INSTITUT FEMTO-ST

UMR CNRS 6174

Assessing power needs to run a workload with quality of service constraint on green datacenters

Louis-Claude Canon — Damien Landré — Laurent Philippe — Jean-Marc Pierson — Paul Renaud-Goud

Research report n° RR-FEMTO-ST-1376

DÉPARTEMENT DISC – March 3, 2023

Assessing power needs to run a workload with quality of service constraint on green datacenters

Louis-Claude Canon , Damien Landré , Laurent Philippe , Jean-Marc Pierson , Paul Renaud-Goud

Département DISC

DEODIS

Research report no RR-FEMTO-ST-1376 March 3, 2023 (33 pages)

Abstract: Datacenters are an essential part of the internet but their continuous development requires finding sustainable solutions to limit their impact on climate change. The Datazero2 project aims to design datacenters running solely on local renewable energy. In this research report we tackle the problem of computing the minimum power demand to process a workload under quality of service constraint. To solve this problem we propose a binary search algorithm that requires the computation of ma- chine configurations with maximum computing power. When machines are heterogeneous, we face the problem of choosing the machines and their DVFS state. A MILP (Mixed-Integer Linear Programming), to find the optimal solutionn, and four heuristics that give satisfactory results in a reasonable time are proposed. The bests reach an average deviation from the optimal solution of 0.03% to 0.65%.

Key-words: Green datacenter , Power consumption , Optimization

FEMTO-ST Institute, DISC research department
UFR Sciences - Route de Gray - F-25030 BESANÇON cedex FRANCE
Tel: (33 3) 81 66 65 15 – Fax: (33 3) 81 66 64 50 – e-mail: fvernill@femto-st.fr

Evaluation de la consommation d'énergie nécessaire à l'exécution d'un workload sous la contrainte d'une qualité de service pour des datacenters verts

Résumé : Les datacenters sont un élément essentiel de l'internet, mais leur développement continuel nécessite de trouver des solutions durables pour limiter leur impact sur le changement climatique. Le projet Datazero2 vise à concevoir des datacenters fonctionnant uniquement avec des énergies renouvelables locales. Dans ce rapport de recherche, nous abordons le problème du calcul de la puissance minimale requise pour traiter un workload sous la contrainte d'une qualité de service. Pour résoudre ce problème, nous proposons un algorithme dichotomique qui nécessite le calcul de la configuration des machines ayant une puissance de calcul maximale. Lorsque les machines sont hétérogènes, nous sommes confrontés au problème du choix des machines et de leur état DVFS. Un MILP (Mixed-Integer Linear Programming), pour trouver la solution optimale, et quatre heuristiques donnant des résultats satisfaisants en un temps raisonnable sont proposés. Les meilleures heuristiques atteignent un écart relatif moyen par rapport à la solution optimale de 0.03% à 0.65%.

Mots-clés : Datacenter vert , Consommation de puissance , Optimisation

FEMTO-ST Institute, DISC research department
UFR Sciences - Route de Gray - F-25030 BESANÇON cedex FRANCE
Tel: (33 3) 81 66 65 15 – Fax: (33 3) 81 66 64 50 – e-mail: fvernill@femto-st.fr

Assessing power needs to run a workload with quality of service constraint on green datacenters

Louis-Claude Canon , Damien Landré , Laurent Philippe , Jean-Marc Pierson

March 3, 2023

Abstract

Datacenters have become an essential part of the internet but their continuous development requires finding sustainable solutions to limit their impact on climate change. The Datazero2 project aims to design datacenters running solely on local renewable energy. Energy demand of datacenter's machines can be optimized to improve the use of energy. In this paper we tackle the problem of computing the minimum power demand to process a workload under quality of service constraint. We propose a binary search algorithm to solve this problem. This algorithm requires a machine configuration that maximizes computing power. In a heterogeneous environment, the difficulty of the problem lies in the choice of the machines to be switched-on and their DVFS (Dynamic Voltage and Frequency Scaling) state. This can be computed by a MILP (Mixed-Integer Linear Programming), but with an important computation time. Four heuristics are proposed to maximize computing power and give satisfactory results in a reasonable time, with an average deviation from optimal solution of 31.84%, 0.12%, 0.65% and 0.03%. These promising results encourage us to use heuristics to address the problem of computing the minimum power demand.

1 Introduction

Since a decade datacenters have become an essential part of the internet, either being at the edge or at the center, and their number and size are continuously increasing, as their global energy consumption. These datacenters represented in 2018 1% of the global energy consumption, that is to say 6% more than in 2010 [17] and it is estimated that these numbers are growing. Indeed, according to GreenIT [6], it is estimated that, by 2025, the number of users will have increased by 1.1 billion, energy consumption will have multiplied by 2.9 and greenhouse gas emissions by 3.1.

To reduce the datacenter impact on climate change several research works propose solutions to optimize their energy consumption [7], [21], [14]. These solutions are essential on the way to efficiency but cannot achieve a drastic reduction of the carbon footprint. Other projects and research works claim to reduce their brown energy consumption [4], [11], [2]. The objective of the Datazero project [24] (2015-2019) and Datazero2 (2020-2024) is to investigate the solutions to design and operate a datacenter only fueled by renewable energies. By design, this project builds on a negotiation [29] between the electrical management (power production and storage) and the IT management (workload processing) to choose a power command that will be applied in the next time window, typically the coming 72 hours. A power command refers to the control commands that are asked to the electrical side, so that the needed power is provided along the time window.

The negotiation is developed as a process based on game theory that loops on asking for IT consumption and power production predictions over the time window to converge after several

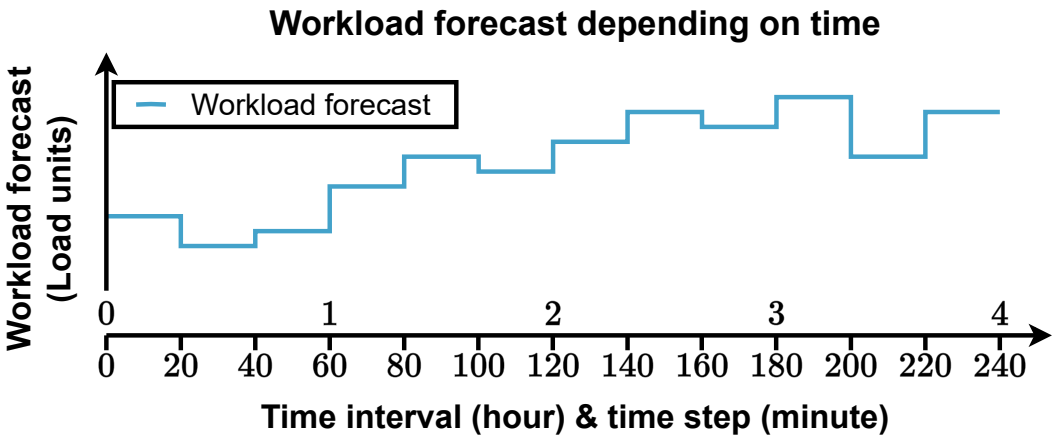


Figure 1: A workload forecast depending on time. Each time interval is composed of several time steps where the workload differs. For each time step is represented the total amount of operations in GFlops to be processed.

iterations on an acceptable solution for both the IT and electrical management. The result of the negotiation is a power command for the time window, represented as a time series of power values for each time interval over this time window, called a power profile. This power command is then applied on the infrastructure.

As a matter of fact, the negotiation need predictions of the power needs during the time window. In this article, the problem we tackle is to compute the minimum power profile required to process a workload forecast. We do not address the problem of workload forecast that has been widely studied already [18]. Rather, we investigate the problem of transforming a workload prediction to an optimized usage of a given infrastructure that minimizes the electrical power needs. Since the negotiation process is interactive, this computation must last a reasonable time [30].

It must be noted already that we consider a consolidated workload, and not individual jobs. Therefore a workload represents the total amount of work units that have to be processed along time. A workload possibly aggregates the work units of several jobs that may concurrently use the infrastructure and share each of the machines of the infrastructure.

Fig. 1, 2 and 3 give a motivating example of the problem. Fig. 1 shows a workload forecast for 4 hours. Note that this workload varies at an higher frequency (a few seconds to a minute) than the electrical power and that time intervals (defined as the minimum duration of the electrical command), are composed of several time steps (*i.e.* 4 time intervals and time steps of 20 minutes in the example).

In order to process this load, it is necessary to provide sufficient computing power, *i.e.* switching-on machines in the datacenter and setting them in an adequate state using DVFS (Dynamic Voltage and Frequency Scaling). In the following, we note configuration the set of states of the machines of the infrastructure, *i.e.* their on/off and DVFS states. In the datacenter the machines may be heterogeneous [31], [25], [26] [27] hence different configurations provide different computing capabilities and have different consumption. Since the power command must be maintained over a time interval, there is no need to change the configuration during a time interval. The problem is hence to compute, for each time interval, a configuration that is able to process the workload. This, in turn, defines the power consumption over the time interval and the power profile over the time window.

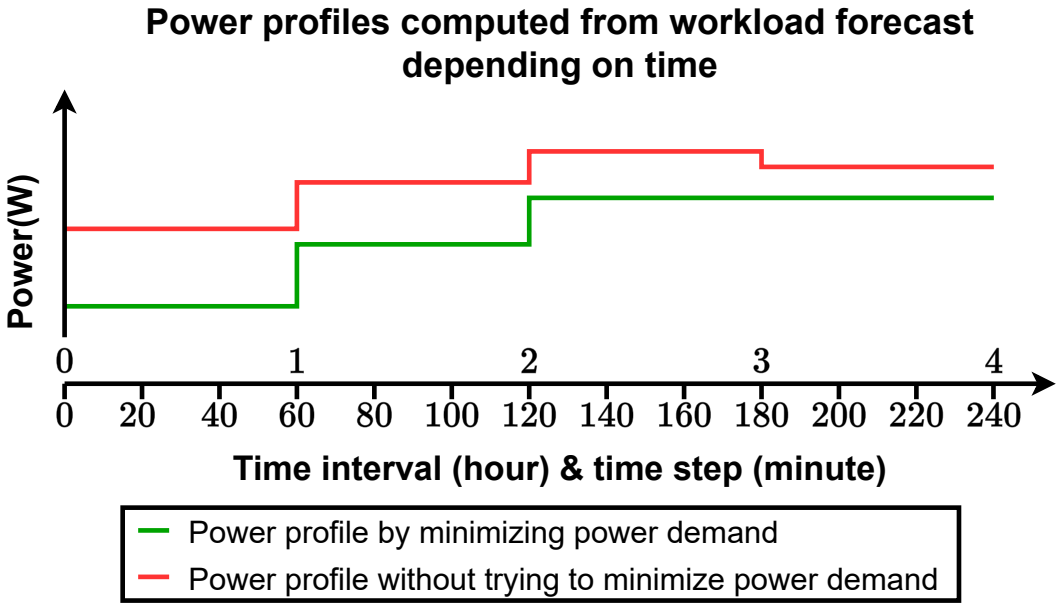


Figure 2: A power profile obtained by minimizing power demand of the machines at each time interval to process the load (green) and a power profile obtained without trying to minimize power demand of the machines at each time interval to process the load (red). It can be seen that the red power profile requires more power than the green power profile.

The red power profile on Fig. 2 is obtained with successive configurations, one for each time interval, that are able to process the workload. It is however preferable to propose a power profile that minimizes the power demand at each time interval in order to save energy and to anticipate periods of under-production in the future, because the energy production forecast is uncertain. The machine configuration can be computed with the objective to reduce power demand while having the necessary computing power. The green power profile is obtained by minimizing power demand at each time interval, *i.e.* finding the less consuming machine configuration whose computing power is sufficient to process the load. This turns out to be a complex optimization problem.

Finally, as the load may be flexible, it is also possible to minimize power needs over a time interval by moving work units, called load parts, over time steps, as shown in Fig. 3, provided that their associated deadline are respected. Note the two y-axis with different units on the figure.

This paper contributes with multiple variants of an algorithm that computes a minimized power profile. The algorithm realizes this computation in steps. The main step iterates on each time interval of the time window. For each interval, a binary search algorithm is used to find a minimized power value. Last, for each power value, the algorithm computes the maximum processing capacity that can be reached using the datacenter machines and tries to schedule the workload under quality of service (QoS) constraints, using the processing capacity, to check if this is feasible. We propose several solutions, a MILP and different heuristics, to compute the maximum computing capacity for a power value.

Section 2 details the related work, while Section 3 describes formally the problem of computing a capping value from a workload forecast and maximizing a processing capacity within a given power value. Section 4 presents algorithms and heuristics while section 5 presents experiments and results. Finally, section 6 summarizes the paper, highlighting the main conclusion.

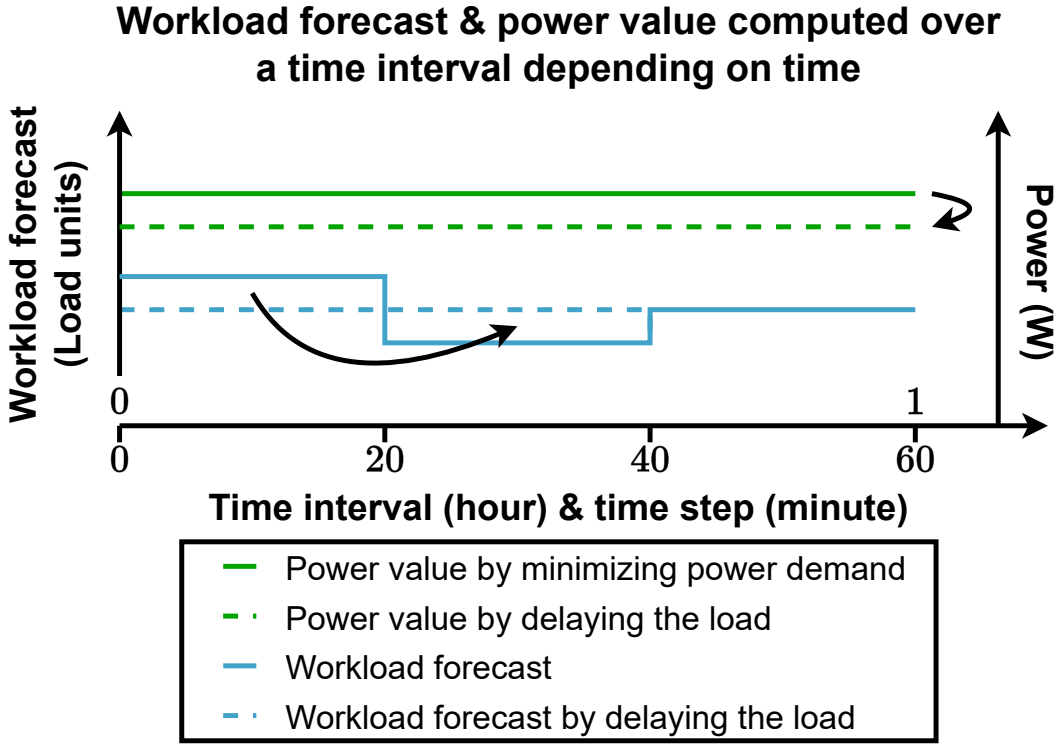


Figure 3: Over a time interval (zoom here on the first time interval of Fig 1 and Fig 2), the workload differs from one time step to another. It is possible to minimize the power value needs over a time interval by moving the load to other time steps.

2 Related Work

In order to minimize energy or power consumption while possibly meeting another criteria, different online or mixed approaches have been considered in the literature.

In [34], an online method manage energy consumption and workload scheduling for hybrid geo-distributed datacenters. Several variables are taken into account such as the variation of the electricity price, the power consumption of the cooling, machines, or constraints on renewable energy. In a similar way, an online Evolutionary Energy Efficient Virtual Machine Allocation (EEE-VMA) algorithm, a metaheuristic based on genetic algorithm to optimize energy consumption, performance degradation as well as power grid cost for multiple hybrid datacenters [23]

Zhang et al. [33] propose PoDD, an online power-capping algorithm to maximize performances of homogeneous servers for dependent application workloads (applications with a front-end and a back-end. The front-end produces the data that are consumed by the back-end). A similar method is also proposed for heterogeneous nodes of a datacenter [9].

In [32], different online algorithms are introduced to minimize the performance degradation and the total energy consumed: LmsReg, based on regression, which detects overloaded servers and migrates virtual machines to another, and MuP to address the trade-off between power consumption, number of migrations, server performance and the total number of servers that have been switched off in the selection of virtual machines. These algorithms migrates virtual machines from over-loaded servers to under-loaded servers. Other methods using virtual machine allocation and migration are proposed [19], [13], [16], [20], [8]. In [15], an online holistic approach schedules virtual machines to minimize the total energy consumption of the datacenter by considering the datacenter as a whole and not trying to divide it into several parts to be treated separately from each other.

In [20], an online multi-objective algorithm optimize energy consumption, by taking into account QoS, energy, number of active servers, number of virtual machine migrations on servers. A similar method in [10] is used by considering DVFS, temperature-dependent task scheduling, dynamic resource provisioning, and cooling management

In [12], in the context of cloud datacenters, a method for prediction of the total energy consumption of the datacenter is proposed to support the datacenter energy management system that controls and coordinates all the equipments. This method evaluates and selects the importance of the variables of all the equipments to make the prediction with application of a PCA to reduce the dimensions of the variables. Then a neural network makes the prediction on the total energy consumption (a single value in a future close to 20 minutes, because it is the time necessary for the system to reach a desired temperature). Finally, an online module is in charge of correcting and updating the model based on the forecast errors.

None of these works addresses the offline minimization of power consumption under the constraint of deadline violations in the case of homogeneous and heterogeneous machines with different amount of work to process.

3 Problem

As previously explained, the problem we face is to compute the minimum power profile, a time series of power values, needed to process a given workload, composed of load parts. In this section, we first define the problem of optimizing the power need necessary to process the workload and maximizing a processing capacity, on a time interval and for a given power value, then formally define the model and the objectives.

3.1 Definition

The power management system of a green datacenter, running solely on renewable energy, needs to plan its power sources and storage usage to correctly fuel the machines of the datacenter. Due to technical constraints [24], the power command delivered to the machines cannot be variable over a minimum time duration, a time interval. A common duration for a time interval ranges from 15 minutes to one hour. To correctly plan the power usage over a time window, we need to define a power profile that gives, for each time interval, a power need value, *i.e.* a constant value, for the datacenter. This power value is the power that will be needed by the datacenter to process the incoming workload. Its computation is based on a workload forecast.

In the context of this paper, we assume that the workload forecast is an input of the problem and that the solution must be able to handle any workload, whatever its characteristics. gives the variation of the load on time steps in the coming time intervals. A time interval is thus subdivided into multiple time steps. Note that, the timings are different between power time intervals and workload time steps. A common duration for a time step ranges from 1 second to one minute. The workload is composed of several load parts, each arriving at a given time step. Since load arriving in the same time step may have different QoS constraints, a deadline is associated to each load part.

To process the workload, we need enough computing power in the infrastructure. This requires to find an appropriate configuration of the datacenter machines (off, on and other DVFS states). Because the machines consume power to process the load, a configuration defines a power need value for a time interval and a series of configurations define in turn a power profile for the coming time window. Since the power supply only relies on intermittent renewable energies, storing as much energy as possible is essential to be able to operate the datacenter during periods of underproduction. We thus intend to save as much power as possible by requesting as little power as possible for each time interval.

Finally we face the problem of minimizing the power value of a time interval given a workload forecast and a set of machines. Note that it is an offline problem scheduling because the workload forecast is available before the computation of the power needs. However, it must be remembered that solving this offline problem must be efficient, because it is repeated for all time intervals of the time window to create a power profile, and the power profile is in turn used in the iterative process of the negotiation (meaning for each time window, several power profiles will be generated and used in the negotiation loop).

To give the users a flowtime guarantee and to avoid too long waiting times that may dissuade them from using the datacenter, we enforce the following constraint: the submitted workload must be totally or partially processed during the considered time interval, according to a deadline violation threshold not to be exceeded.

3.2 Model

We consider the workload forecast to be discretized in time steps. Formally, we denote by T the number of time steps, and we normalize the time axis such that the t^{th} time step begins at time $t-1$, for $t \in \mathcal{T} = \{1, \ldots, T\}$. We define Δt as the duration of a time step in seconds. At each time step, we assume that the workload is composed of several load parts. We define the total workload as a set of W load parts, l_k for $k \in \mathcal{W} = \{1, \ldots, W\}$. For instance, on the first time step of Fig. 4, the forecast is composed of two load parts, l_1 and l_2. A load part l_k is defined by its release time r_k (*i.e.* the time step when load part l_k arrives), its amount of operations to be processed p_k and a deadline d_k. Note that the model concentrate on the computing power needs whereas its consumption can vary depending whether the workload is CPU, I/O, memory or network intensive. On the other the cpu consumption is the main part of the load consumption.

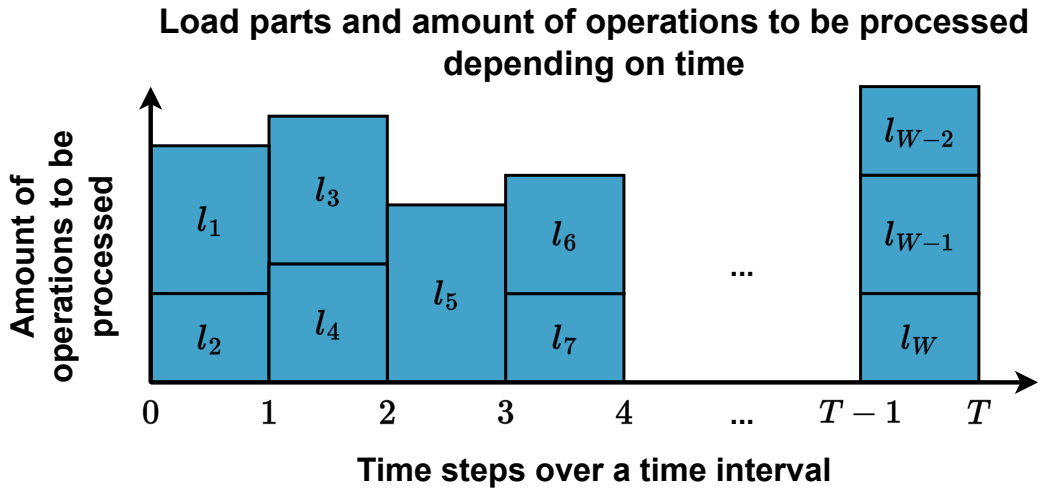


Figure 4: Load parts and amount of operations of a workload forecast.

On Fig. 4, p_1 (the amount of operations of l_1) is two times larger than p_2. The deadline d_k is defined as a duration. A load part l_k, which arrives at the time step $t = r_k$ with a deadline d_k must be finished no later than $r_k + d_k$. All the operations of a load part l_k are restricted to the same deadline d_k. For instance, in Fig. 4 if load part l_1 has a deadline of two time steps, then it can be delayed and processed partially or completely on time steps 1, 2 and/or 3.

The load is processed by M machines of a datacenter, which are noted machine i with $i \in \mathcal{M} = \{1, \ldots, M\}$. Considering the power consumption, machine i dissipates a power $static_i$ when it is idle. Each machine can be set in S_i different DVFS states [5]. A DVFS state of a machine is noted $j \in \mathcal{S}^{(i)} = \{0, \ldots, S_i\}$. The set of machines with their DVFS states defines the configuration of the datacenter. We note $\mathcal{S}$ the set of DVFS states of all machines, $\mathcal{S} = \{\mathcal{S}^{(1)}, \ldots, \mathcal{S}^{(M)}\}$.

A DVFS state j defines $g^{(i)}_{\max_j}$ the maximum amount of operations per second that the machine i can process and $power^{(i)}_j$ the consumed power per operation per second of the machine. For the sake of simplicity, we consider an average value for this consumed power. The model could be extended to consider different power consumption for different operations. If machine i is switched-on, it computes $g^{(i)}$ operations per second with $0 \leq g^{(i)} \leq g^{(i)}_{\max_j}$ while dissipating $power^{(i)}_j$ power per operation per second. Therefore, if a machine i computes several load parts l_k in state j during a time Δt in seconds with an amount of operations to be processed $g^{(i)}\Delta t = \sum_{k \in \mathcal{W}} p_k \leq g^{(i)}_{\max_j}\Delta t$, it consumes a power of $static_i + g^{(i)} \times power^{(i)}_j$. We assume that when a machine is off, its DVFS state is $j = 0$ and it does not consume any power, nor does it process any operation.

3.3 Objectives

The main objective is to propose a power profile, which minimizes the power value needs on each time interval. Each power value must be sufficient to process the workload and to respect the constraint on violated deadlines, but not excessive in order to save energy over the time interval. As previously said, the power value has to be constant over one time interval: Minimizing the power value on one time interval is therefore our objective. The computation is then repeated for each interval.

The inputs of the problem are the characteristics of the machines and the workload forecast. In order to respect the constraint on the number of violated deadlines, the maximum ratio of violated deadline, noted $D_{\max}$, is fixed. The output of the problem is the power value for the interval, *i.e.* the criterion to minimize.

Solving this problem implies to decide the configuration of the machines and the schedule of the load parts on the time steps. In the following we define the decision variables and their constraints, first for the machines and then for the workload forecast.

We define the decision variable $x_{i,j}$ to determine the machines to be switched-on or -off and their DVFS state. For each machine i and for each DVFS state j, $x_{i,j} = 1$ if the DVFS state j of machine i is selected, otherwise $x_{i,j} = 0$ (1). These variable hence define the machine configuration. For a machine, we consider that only one DVFS state can be selected and remains the same for the entire duration of the time interval (2). Note that in an heterogeneous machine environment, *i.e.* with different type of machines, the difficulty lies in the choice of the machines to be switched-on and the choice of their DVFS state.

$$\forall (i,j) \in \mathcal{M} \times \mathcal{S},\ x_{i,j} \in \{0,1\} \tag{1}$$

$$\forall i \in \mathcal{M},\ \sum_{j \in \mathcal{S}^{(i)}} x_{i,j} = 1 \tag{2}$$

We define the decision variable $g^{(i)}$ as the computing power. This computing power is bounded by maximum computing power that can be reached by the machine in DVFS state j (3). Note that the maximum available computing power depends on the choice of the DVFS state of the machine via the decision variable $x_{i,j}$ previously defined.

$$\forall i \in \mathcal{M},\ 0 \leq g^{(i)} \leq g^{(i)}_{\max_j} \tag{3}$$

For each machine i, after determining its DVFS state and its computing power $g^{(i)}$, we can deduce its power consumption P_i knowing its static power $static_i$ and its dynamic power *i.e.* the power consumption per operation at the chosen DVFS state (4).

$$\forall i \in \mathcal{M},\ P_i = \begin{cases} static_i + g^{(i)} \times power_j^{(i)} & \text{if } j > 0 \\ 0 & \text{if } j = 0 \end{cases} \tag{4}$$

The total power consumption P of the machines is then the sum of the power consumption of all machines (5).

$$\sum_{i \in \mathcal{M}} P_i = P \tag{5}$$

We deduce the maximum available computing power $w^{(p)}$, by summing the computing power of all the machines over a time step (6).

$$w^{(p)} = \sum_{i \in \mathcal{M}} g^{(i)} \Delta t \tag{6}$$

It is then necessary to schedule load parts on active machines. For each time step, the total amount of operations to be processed is the sum of the operations to be processed of all load parts of the time step. Knowing the maximum available computing power $w^{(p)}$ (6) over the entire time interval and knowing the duration of a time step, we can determine the total number of operations processed by the machines during a time step. If the total number of operations to process during time step t cannot be fully processed by the machines during a

time step, the remaining operations are delayed and/or killed according to the deadlines. The total amount of operations killed is noted *opk* and it is a decision variable of the problem.

The ratio of violated deadlines D is defined as the amount of operations killed over the amount of operations to be processed during the time interval (7). This ratio must not exceed the maximum ratio of violated deadlines $D_{\max}$ given as input (8).

$$D = \frac{opk}{\sum_{k \in \mathcal{W}} p_k} \tag{7}$$

$$D \leq D_{\max} \tag{8}$$

4 Determining the minimum power value

In this section, we propose a solution to the problem of minimizing the power value for a time interval under the deadline violation constraint. A binary search algorithm is proposed to determine the minimum power value to request. For a given power value, the algorithm first computes a machine configuration and then schedules the workload to determine the amount of processed, delayed and killed operations in the time interval. The binary search algorithm is then repeated for each time interval of the time window to generate a power profile.

4.1 Binary search algorithm

In this section, we detail the binary search approach to solve the problem of minimum requested power value (Algorithm 1).

Algorithm 1: Binary search algorithm to minimize the power need to run a workload on machines under the constraint of a deadline violation ratio

Data: $\mathcal{M}, \mathcal{S}, \mathcal{W}, \mathcal{T}, \epsilon, D_{\max}, \Delta t$
Result: minimize P

begin
  $P_{\min} \leftarrow 0$
  $P_{\max} \leftarrow \sum_{i \in \mathcal{M}} (static_i + power^{(i)}_{S_i} \times g^{(i)}_{\max_{S_i}})$
  while $P_{\max} - P_{\min} \geq \epsilon$ **do**
    $P \leftarrow (P_{\min} + P_{\max})/2$
    $w^{(p)} \leftarrow config(\mathcal{M}, \mathcal{S}, P, \Delta t)$
    $opk \leftarrow 0$
    $\bar{\mathcal{W}} \leftarrow \mathcal{W}$
    for $t \in \mathcal{T}$ **do**
      $\bar{w}^{(p)} \leftarrow w^{(p)}$
      $\bar{\mathcal{W}}, opk \leftarrow schedule(\bar{\mathcal{W}}, opk, \bar{w}^{(p)}, t)$
    $D \leftarrow opk / \sum_{k \in \mathcal{W}} p_k$
    if $D \leq D_{\max}$ **then** $P_{\max} \leftarrow P$ **else** $P_{\min} \leftarrow P$

To initiate the dichotomy, we set the maximum power $P_{\max}$ to the case where all the machines are used to their maximum capacity. The minimum power $P_{\min}$ is initialized to 0. We define ϵ as the stopping criterion of the algorithm. At each iteration the algorithm computes with the *config* function the maximum available computing power $w^{(p)}$ of the machine configuration under the power value constraint P of the current iteration. The *schedule* function is then run

for each time step of the time interval to determine the schedule and the value of *opk*, the number of killed operations, which is used to calculate the ratio of violated deadlines D over the time interval when the P value is used. Note that the workload $\mathcal{W}$ is copied (line 8, Algorithm 1) because it needs the original set at each iteration and because the *schedule* function needs to modify it (to potentially delay or even kill load parts when deadlines are exceeded).

If the ratio of violated deadlines D exceeds the threshold $D_{\max}$, it means that the computing power is not sufficient, meaning that the current power value P is not enough. It is hence necessary to iterate with a higher value of P to increase the computing power with a more powerful machine configuration. Conversely, if the ratio of violated deadlines D does not exceed the threshold $D_{\max}$ then the power P can be decreased.

In the following, several versions of the *config* function will be given in Section 4.2), and the *schedule* function is given in Section 4.3.

4.2 Maximizing computing power

The power supplied by the binary search algorithm is used to determine the machine configuration. The simplest case is to consider homogeneous machines with only two DVFS states (switch-on or -off) since it is enough to calculate how many machines can be powered with the given power value to provide the most computing power. If homogeneous machines have several DVFS-states there is already a decision to take between switching-on a new machine and putting it in a higher DVFS-state. In the heterogeneous case several configurations are possible for a given power, but all of them do not provide the same computing power. It is therefore important to improve the power efficiently by determining an optimal machine configuration.

In the following sections, we consider heterogeneous machines with multiple DVFS states. Note that this includes the homogeneous case with the same DVFS states on all machines. The problem of computing the maximum computing power $w^{(p)}$ with heterogeneous machines is a generalization of the knapsack problem and is hence NP-Complete. The proof is based on the definition of a particular case of the problem where the computing $g^{(i)}$ allocated to a machine can only be 0 or $g^{(i)}_{\max_j}$. Note that, in the general case, the $g^{(i)}$ are coded in a discrete variable that ranges from 0 to $g^{(i)}_{\max_j}$. In this particular case, we just give the lowest possible value to $g^{(i)}_{\max_j}$ and, hence, $g^{(i)}$ has only two possible values, *i.e.* 0 or $g^{(i)}_{\max_j}$. In that case, the power consumed by a computing machine becomes constant, $P_i = static_i + g^{(i)}_{\max_j}$ (with $static_i = 0$ in our case). This correspond to an instance of the knapsack problem where the items are the machines, their weights are the consumed power P_i and their values are the maximum computing power $g^{(i)}_{\max_j}$. Last the capacity of the knapsack is the total power available P and selecting $g^{(i)}_{\max_j}$ for the computing $g^{(i)}$ is equivalent to selecting the corresponding item. Since this problem is NP-Complete, we first designed a MILP (Mixed-Integer Linear Programming). We then propose different heuristics to address this problem.

4.2.1 Mixed Integer Linear Programming

The MILP is described by (9). The objective function is to maximize the computing power of the machines.

$$\begin{cases} & \text{maximize } \sum_{i=1}^{M} g^{(i)} \\ s.t.: & \\ \sum_{j=0}^{S_i} x_{i,j} & = & 1 \\ g^{(i)} & \leq & \sum_{j=0}^{S_i} x_{i,j} \times g^{(i)}_{\max_j}) \\ P_i & = & \sum_{j=1}^{S_i} x_{i,j}(static_i + g^{(i)} power^{(i)}_j) \\ \sum_{i=1}^{M} P_i & \leq & P \end{cases} \tag{9}$$

Under the following constraints.

$$\begin{cases} \forall i \in \mathcal{M}, \forall j \in \mathcal{S}^{(i)} & x_{i,j} \in \{0,1\} \\ \forall i \in \mathcal{M} & g^{(i)} \geq 0 \\ \forall i \in \mathcal{M} & P_i \geq 0 \end{cases}$$

Using the binary decision variable $x_{i,j}$, the first constraint states that a machine, for all $i \in \mathcal{M}$, must have a single DVFS state j among all possible DVFS states of the machine from 0 to S_i (including the switched-off state $j = 0$). Depending on the selected DVFS state we need to determine, for all $i \in \mathcal{M}$, the computing power in the second constraint, which must not exceed the maximum computing capacity of the machine. Then, knowing the DVFS state and the computing power of the machine, the third constraint bounds the power consumption of the machine, for all $i \in \mathcal{M}$. Finally, the fourth constraint imposes that the total power consumption of the machines must not exceed the power value fixed by the binary search algorithm (Algorithm 1) during an iteration.

As shown by the experiments presented later in section 5, the MILP calculation takes 2.83 seconds in average sec, but up to 30 seconds in complex cases. This calculation has to be repeated for each iteration of the binary search algorithm and the binary search is used for each time interval so that the running time may reach up to more than one hour. As previously explained the power profile is used in the negotiation process to anticipate the power which, in turn, makes several iterations before taking a decision. Although based on offline calculations, it is hence used in an interactive process for which waiting one hour for a proposition does not make sens. For this reason we propose in the following some heuristics that can provide solutions in a shorter time.

4.2.2 Random Choice heuristic

A first trivial heuristic proposal is to randomly choose the type of machine to switch-on. When a machine is switched-on, it is allocated the power needed to provide the maximum computing power. The DVFS state chosen is the one maximizing the computing power according to the remaining power. This step is repeated until the power is insufficient and/or there are no more machines to switch-on. The advantage of this heuristic is its fast execution time but it provides unsatisfactory results compared to the other heuristics presented in the following in the heterogeneous case.

4.2.3 Balance Power-Performance heuristic

The BPP heuristic evaluates the most suitable machines and its DVFS states to switch-on and to choose respectively according to two metrics, which are computing power and performance ratio. This heuristic is very efficient in the homogeneous and heterogeneous case with a very satisfactory execution time.

The Balance Power-Performance heuristic (BPP, Algorithm 2) computes for each machine type and for each DVFS state a normalized score depending on a given power, the machine environment $\mathcal{M}$ and $\mathcal{S}$ and an α parameter. Δt is the length of a time step. BPP switches-on the machine with the highest score. The α parameter (with $0 \leq \alpha \leq 1$) given as input control the trade-off between computing power and performance ratio (ratio power over computing power).

Algorithm 2: The *BPP* function for a fixed alpha value to generate a machine configuration

Data: $\mathcal{M}, \mathcal{S}, P, \alpha, \Delta t$
Result: Total available computing power $w^{(p)}$

1 **begin**
2 $w^{(p)} \leftarrow 0$
3 $enoughPower \leftarrow True$
4 $machineUse \leftarrow 0$
5 $\bar{\mathcal{M}} \leftarrow \mathcal{M}$
6 **while** $enoughPower$ ***and*** $machineUse \neq M$ **do**
7 $\mathcal{R} \leftarrow computeRatio(\bar{\mathcal{M}}, \mathcal{S}, P)$
8 **if** $\mathcal{R} \neq \emptyset$ **then**
9 **for** $\{(i,j), ratio_{i,j}, g^{(i)}\} \in \mathcal{R}$ **do**
10 $score_{i,j} \leftarrow \alpha(g^{(i)} / \max_{(k,l)\in\mathcal{R}}(g^{(k)})) + (1-\alpha)(\min_{(k,l)\in\mathcal{R}}(ratio_{k,l})/ratio_{i,j})$
11 $k, l \leftarrow \arg\max_{(i,j)\in\mathcal{R}}(score_{i,j})$
12 $w^{(p)} \leftarrow w^{(p)} + g^{(k)}\Delta t$
13 $P \leftarrow P - (static_k + g^{(k)} power_l^{(k)})$
14 $\bar{\mathcal{M}} \leftarrow \bar{\mathcal{M}} \backslash \{k\}$
15 $machineUse \leftarrow machineUse + 1$
16 **else** $enoughPower \leftarrow False$

The computing power criteria of a machine is chosen since the objective is to maximize the total computing power of the machines $w^{(p)}$. The performance ratio criteria of a machine is chosen to minimize the power consumed per unit of computing power. This corresponds to the total power consumed by the machine over the provided computing power. The ratio hence represents the machine's power consumption per unit of computing power (10).

$$\forall (i,j) \in \mathcal{M} \times \mathcal{S},\ ratio_{i,j} = \frac{static_i + g^{(i)} power_j^{(i)}}{g^{(i)}} \tag{10}$$

Both criteria are computed beforehand with the *computeRatio* function (Algorithm 3) and aggregated in the same computation of a machine score (line 10 of Algorithm 2). Both criteria are used because switching-on machines in the DVFS state with the most computing power does not necessarily give an optimal machine configuration. On the other hand, if the power allows it, it is better to switch-on all the machines in the higher DVFS state that has not necessarily the best performance ratio.

The nearer alpha is to 0, the more weight is given to the computing power produced in the choice of the machine to be switched-on. Inversely, the closer alpha is to 1, the more weight is given to the performance ratio. Depending on the alpha parameter value, the configurations proposed by BPP can be different. For this reason, several alpha values are assessed in order

Algorithm 3: The *computeRatio* function that computes for each machine and DVFS state the performance ratio and computing power

Data: $\bar{\mathcal{M}}, \mathcal{S}, P$
Result: $\mathcal{R}$
begin
  $\mathcal{R} \leftarrow \emptyset$
  for $i \in \bar{\mathcal{M}}$ **do**
    if $P > static_i$ **then**
      for $j \in \mathcal{S}^{(i)}$ **do**
        $g^{(i)} \leftarrow \min((P - static_i)/power_j^{(i)}, g_{\max_j}^{(i)})$
        $ratio_{i,j} \leftarrow (static_i + g^{(i)} power_j^{(i)})/g^{(i)}$
        $\mathcal{R} \leftarrow \mathcal{R} \cup \{(i,j), ratio_{i,j}, g^{(i)}\}$

to produce different machine configurations and the algorithm returns the one maximizing the total computing power (*config* function with BPP, Algorithm 4).

Algorithm 4: The *config* function for Balance Power-Performance heuristic by running different alpha values and keep the best machine configuration

Data: $\mathcal{M}, \mathcal{S}, P, \Delta t, n$
Result: Total available computing power $w^{(p)}$
begin
  $w^{(p)} \leftarrow 0$
  for $\alpha \leftarrow 0$ *to* 1 *by* $1/(n-1)$ **do**
    $\bar{w}^{(p)} \leftarrow BPP(\mathcal{M}, \mathcal{S}, P, \alpha, \Delta t)$
    if $\bar{w}^{(p)} > w^{(p)}$ **then** $w^{(p)} \leftarrow \bar{w}^{(p)}$

4.2.4 Best State Redistribute Without Static heuristic

The BSRWS heuristic focuses on the performance ratios of the machines. The advantage of this heuristic is its accuracy with a satisfactory execution time, but which increases depending on power. In the heterogeneous case, some solutions deviatefrom the optimal because it switches on too many machines.

The *config* function of Best State Redistribute Without Static heuristic (BSRWS, Algorithm 5) switches-on as much machines with the best performance ratio as it is possible without exceeding the given power P (Algorithm 6). The *switchOn* function (Algorithm 6) is called to determine machines to be switched-on $\mathcal{M}^{(a)}$, the remaining power available after switching-on these machines and the total computing power $w^{(p)}$ provided by these machines. As for Algorithm 2, the *computeRatio* function (Algorithm 3) is used to compute beforehand the performance ratio of each machine for each DVFS states.

Then, if no more machine can be switched-on and there is power left, either because all the machines are on or because there is not enough power to switch-on more machines, the remaining power is redistributed to the switched-on machines. This redistribution allows to increase the DVFS state of the switched-on machines and thus their computing power. It is done by the *redistribute* function (Algorithm 7). As long as it is possible to redistribute power

Algorithm 5: The *config* function for Best State Redistribute Without Static heuristic to generate a machine configuration

Data: $\mathcal{M}, \mathcal{S}, P, \Delta t$
Result: $w^{(p)}$
begin
  $\mathcal{M}^{(a)}, P, w^{(p)} \leftarrow switchOn(\mathcal{M}, \mathcal{S}, P, \Delta t)$
  if $\mathcal{M}^{(a)} \neq \emptyset$ ***and*** $P > 0$ **then**
    $w^{(p)} \leftarrow redistribute(\mathcal{M}^{(a)}, \mathcal{S}, P, \Delta t, w^{(p)})$

Algorithm 6: The *switchOn* function that switch-on machines with the best performance ratio

Data: $\mathcal{M}, \mathcal{S}, P, \Delta t$
Result: $\mathcal{M}^{(a)}, P, w^{(p)}$
begin
  $w^{(p)} \leftarrow 0$
  $\bar{\mathcal{M}} \leftarrow \mathcal{M}$
  $\mathcal{M}^{(a)} \leftarrow \emptyset$
  $enoughPower \leftarrow True$
  while $enoughPower$ ***and*** $\bar{\mathcal{M}} \neq \emptyset$ **do**
    $\mathcal{R} \leftarrow computeRatio(\bar{\mathcal{M}}, \mathcal{S}, P)$
    if $\mathcal{R} \neq \emptyset$ **then**
      $k, l \leftarrow \arg\min_{(i,j)\in\mathcal{R}}(ratio_{i,j})$
      $w^{(p)} \leftarrow w^{(p)} + g^{(k)}\Delta t$
      $P \leftarrow P - (static_k + g^{(k)} power_l^{(k)})$
      $\bar{\mathcal{M}} \leftarrow \bar{\mathcal{M}} \backslash \{k\}$
      $\mathcal{M}^{(a)} \leftarrow \mathcal{M}^{(a)} \cup \{(k, l)\}$
    else $enoughPower \leftarrow False$

to increase the computing power of the machine configuration, the new computing power $\bar{g}^{(i)}$ is calculated for each switched-on machine depending on the available power. For each machine, the computing power of each higher DVFS state of the machine is calculated. Then the machine and the DVFS state with the best performance ratio $power_j^{(i)}$ is chosen. The DVFS state of the chosen machine is then increased to provide a higher computing power.

Algorithm 7: *redistribute* function that allocates surplus power to switched-on machines

Data: $\mathcal{M}^{(a)}, \mathcal{S}, P, \Delta t, w^{(p)}$
Result: $w^{(p)}$
begin
  $redistribution \leftarrow True$
  while $redistribution$ **do**
    $\mathcal{R} \leftarrow \emptyset$
    for $(i,j) \in \mathcal{M}^{(a)}$ **do**
      for $k \leftarrow j+1$ *to* S_i **do**
        $\bar{g}^{(i)} \leftarrow \min((P + g^{(i)} \times power_j^{(i)})/power_k^{(i)}, g^{(i)}_{\max_k})$
        if $\bar{g}^{(i)} > g^{(i)}$ **then** $\mathcal{R} \leftarrow \mathcal{R} \cup \{(i,k), \bar{g}^{(i)}\}$
    if $\mathcal{R} \neq \emptyset$ **then**
      $\{(l,m), \bar{g}^{(l)}\} \leftarrow \arg\min_{(i,k)\in\mathcal{R}}(power_k^{(i)})$
      $P \leftarrow P + g^{(l)} power_j^{(l)} - \bar{g}^{(l)} power_m^{(l)}$
      $w^{(p)} \leftarrow w^{(p)} + (\bar{g}^{(l)} - g^{(l)})\Delta t$
    else $redistribution \leftarrow False$

4.2.5 Best State Redistribute Without Static And Removing heuristic

The BSRWS-AR heuristic focuses on the performance ratios of the machines and explore more machine configurations. The advantage of this heuristic is its accuracy compared to BSRWS In the homogeneous and heterogeneous case. However, its execution time is much higher and increases strongly with power. This is due to several machine configurations being explored

The Best State Redistribute Without Static And Removing heuristic (BSRWS-AR, algorithm 8) is a BSRWS heuristic run several times with, at each iteration, removing an available machine (whatever the type of machine, it is the number of available machines that is affected) in order to test configurations with fewer switched-on machines but more power redistributed. The execution stops when the computing power of the found configuration is lower than the previous one. The idea is to evaluate different configurations with less machines. More power is allocated to the remaining switched-on machines which allows to increase their DVFS state and thus their computing power.

4.3 Scheduling the workload on the chosen configuration

Knowing the workload forecast $\mathcal{W}$, the total computing power of the machines and time step t, the *schedule* function (Algorithm 9) schedules the arriving or delayed load parts at this time step. The operations of a load part can be completely or partially processed during the time step. If they are partially processed, the remaining operations are either delayed or killed depending

Algorithm 8: The *config* function with Best State Redistribute Without Static And Removing heuristic

Data: $\mathcal{M}, \mathcal{S}, P, \Delta t$
Result: $wfinal^{(p)}$
begin
  $continue \leftarrow True$
  $wfinal^{(p)} \leftarrow 0$
  while *continue* **do**
    $\mathcal{M}^{(a)}, P, w^{(p)} \leftarrow switchOn(\mathcal{M}, \mathcal{S}, P, \Delta t)$
    if $\mathcal{M}^{(a)} \neq \emptyset$ ***and*** $P > 0$ **then**
      $w^{(p)} \leftarrow redistribute(\mathcal{M}^{(a)}, \mathcal{S}, P, \Delta t, w^{(p)})$
    if $w^{(p)} > wfinal^{(p)}$ **then**
      $wfinal^{(p)} \leftarrow w^{(p)}$
      $M \leftarrow M - 1$
    else $continue \leftarrow False$

on the load part deadline. For this reason, and in order to limit the number of operations killed, load parts with the earliest deadlines are processed first (EDF, Earliest Deadline First).

Algorithm 9: The *schedule* function based on a earliest deadline first scheduling policy

Data: $\bar{\mathcal{W}}, opk, \bar{w}^{(p)}, t$
Result: $\bar{\mathcal{W}}, opk$
begin
  $\mathcal{W}^{(p)} \leftarrow \emptyset$
  for $k \in \bar{\mathcal{W}}$ **do**
    if $r_k \leq t$ ***and*** $r_k + d_k \geq t$ **then** $\mathcal{W}^{(p)} \leftarrow \mathcal{W}^{(p)} \cup \{k\}$
  while $\mathcal{W}^{(p)} \neq \emptyset$ **do**
    $k \leftarrow \arg\min_{l \in \mathcal{W}^{(p)}}(r_l + d_l - t)$
    if $p_k \leq \bar{w}^{(p)}$ **then**
      $\bar{w}^{(p)} \leftarrow \bar{w}^{(p)} - p_k$
      $\mathcal{W}^{(p)} \leftarrow \mathcal{W}^{(p)} \backslash \{k\}$ **and** $\bar{\mathcal{W}} \leftarrow \bar{\mathcal{W}} \backslash \{k\}$
    else
      if $r_k + d_k > t$ **then**
        $p_k \leftarrow p_k - \bar{w}^{(p)}$
        $\bar{w}^{(p)} \leftarrow 0$
        $\mathcal{W}^{(p)} \leftarrow \mathcal{W}^{(p)} \backslash \{k\}$
      else
        $opk \leftarrow opk + p_k - \bar{w}^{(p)}$
        $\bar{w}^{(p)} \leftarrow 0$
        $\mathcal{W}^{(p)} \leftarrow \mathcal{W}^{(p)} \backslash \{k\}$ **and** $\bar{\mathcal{W}} \leftarrow \bar{\mathcal{W}} \backslash \{k\}$

The *schedule* function first initializes the set of load parts $\mathcal{W}^{(p)}$ that can be processed during the current time step given as input. The relevant load parts are those with an arrival time earlier than the current time step and whose deadline is not passed. Then the function processes the load parts with the earliest deadline. A load part is considered to be completely processed if the computing power $\bar{w}^{(p)}$ is sufficient to completely process it. Otherwise, it is partially processed and the rest of the operations to be processed are either delayed ($r_k + d_k > t$) or killed ($r_k + d_k = t$). If these operations are killed, they are added to the *opk* variable that sums the total number of killed operations.

5 Experiment and Results

For our experiment, we consider a medium sized datacenter of 266.7 kW [24] and 10 machine types. Note that, due to the paper length constraint, we concentrate our experiments on only this example of 266.7 kW datacenter but different size of datacenter are experimented and given in the research report to completely assess our heuristics. The machine types are taken from the GRID5000 platform[1].

We implemented inline in Python and run[2] the MILP and the heuristics with input data by simulating with 1241 machines [3] divided into the 10 types. Note that no workload is used for the experiment as we evaluate the solutions of the heuristics compare to the MILP in the determination of the computing power. The characteristics of the machines used are shown in Table 1 and their performance and consumption data are known in advance [22], [28]. For each type of machine, the table gives: the number of machines used for the simulation, the number of DVFS states, the static power *static* when it is idle, the maximum computing power that the machine is able to deliver and the maximum consumed power per operation per second. Table 2 gives an example of more precise data for the Taurus and Gros machines and gives for each DVFS state: the CPU frequency (GHz), the maximum computing power per second ($g_{\max_j}$) and the consumed power per operation per second ($power_j$). Gros machines have better performance ratio than Taurus machines for two main reasons: they have a lower static power and they have a better performance ratio, for a fixed DVFS state, as shown in the table. Thus, based on the performance ratio, Gros machines are advantageous in the choice of the machines to be switched-on and for power redistribution, compared to the Taurus machines. All these data are based on experiments on Grid5000 performed by the ANR project ENERGUMEN [1].

Fig. 5 shows the best performance ratio of the 10 machine types in W/GFlops depending on power, taking into account static power of the machines and all the DVFS states (10). Note that the Gemini machines, which have the highest static power, are located on the right figure. The other machines, which have more or less equivalent static power, are grouped on the left. The lower the static power of a machine and the better the performance ratio, the more advantageous it is to switch-on this machine, depending on power. Also note that some performance ratios of machine types meet others.

Fig. 6 and 7 shows the maximum computing power given by the MILP and heuristics for different power values, from 63 W (the minimum power required to switch-on a machine) to 267 064 W (the maximum power that can be required by all machines at their maximum capacity) by steps of 100 W. This computing power is computed from the datacenter machine configuration. Note that the RC heuristic (Random Choice) is run 100 times for each power value, to reduce the effects of the random choice, and for each power value the average comput-

[1] `https://www.grid5000.fr`

[2] Experiments run on Ubuntu 22.04.1 LTS, Intel Core i7-11850H processor, 32.0 Go of memory, Python 3.10 and PulP 2.6.0 with Gurobi solver.

Table 1: Machine types and their characteristics used for simulation

Machine name	No.	No. of states	Static power (W)	$\max(g_{\max_j})$ (GFlops)	$\max(power_j)$ (W/GFlops)
Taurus	150	13	93.0	220.80	0.38
Parasilo	169	12	94.1	614.40	0.12
Graoully	145	14	98.2	614.40	0.15
Gros	195	14	62.9	633.60	0.12
Grimoire	134	14	121.2	614.40	0.13
Chifflet	89	14	198.7	1075.20	0.09
Grele	106	12	163.2	844.80	0.11
Gemini	34	12	740.8	1408.00	0.09
Graphite	91	10	226	256.00	0.26
Orion	128	13	121.2	220.80	0.39

Table 2: DVFS states and characteristics of Taurus and Gros machines

	Taurus			**Gros**		
j	Freq. (GHz)	$g_{\max_j}$ (GFlops)	$power_j$ (W/GFlops)	Freq. (GHz)	$g_{\max_j}$ (GFlops)	$power_j$ (W/GFlops)
0	0	0	0	0	0	0
1	1.2	116.64	0.24	1.0	287.38	0.12
2	1.3	126.70	0.26	1.1	311.90	0.11
3	1.4	136.02	0.27	1.2	334.81	0.11
4	1.5	145.51	0.28	1.3	368.42	0.10
5	1.6	155.96	0.29	1.4	402.93	0.10
6	1.7	165.44	0.31	1.5	431.19	0.09
7	1.8	174.22	0.32	1.6	460.51	0.10
8	1.9	185.08	0.33	1.7	487.06	0.10
9	2.0	194.79	0.35	1.8	517.93	0.10
10	2.1	201.52	0.37	1.9	546.98	0.09
11	2.2	214.00	0.37	2.0	570.80	0.09
12	2.3	220.80	0.38	2.1	574.01	0.10
13	-	-	-	2.2	633.60	0.12

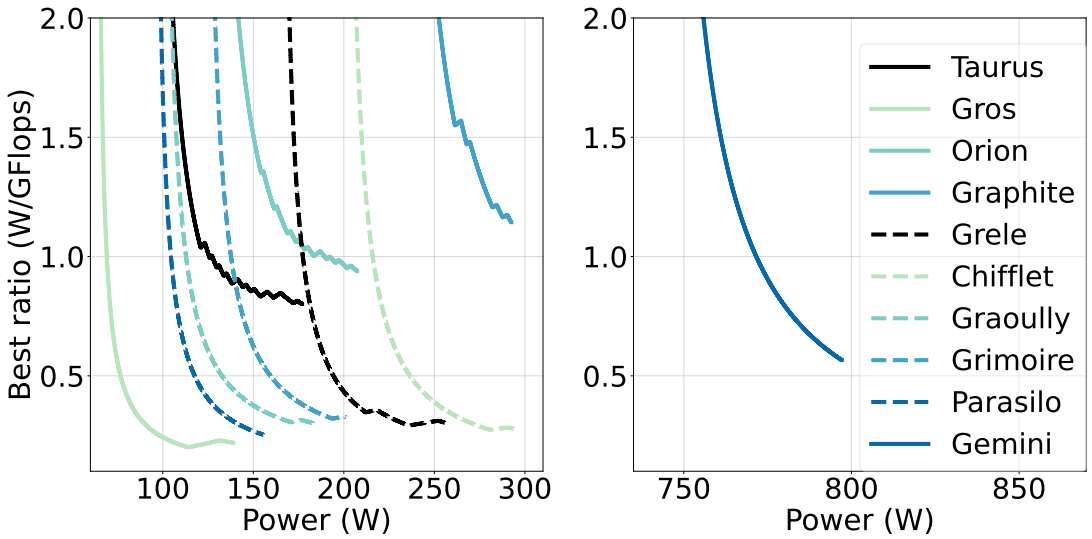


Figure 5: Best performance ratio, by considering static power and all the DVFS states, in W/GFlops.

ing power is shown on the figure. The minimum and maximum computing power are shown in red.

The RC heuristic (Random Choice) significantly deviates from the optimal solution with an average deviation of 31.84%. Since the choice of the machine type to switch-on is random, the RC heuristic may switches-on the least efficient machines, which explains this deviation. This occurs mostly in the heterogeneous case, according to our different experiences. The BPP heuristic (Balance Power-Performance) with alpha from 0 to 1 by steps of 0.05 for each power value and BSRWS-AR heuristic (Best State Redistribute Without Static and Removing) are the closest to the optimal solution. Their average deviation from the optimal is 0.12% and 0.03% respectively. Note that the BSRWS-AR heuristic performs better than the BSRWS (Best State Redistribute Without Static) heuristic since it explores more configurations. Indeed, from 150 kW to 260 kw, the deviation from the optimal is more significant for BSRWS (Fig. 7). The BSRWS heuristic has an average deviation of 0.65%. But according to our different experiences, this is not always the case. Also, BPP outperform BSRWS and BSRWS-AR in other cases of heterogeneity. Note that from approximately 260 kW, the BSRWS heuristic reduces its deviation from the optimal because there is enough power to switch-on all the machines and redistribute the remaining power to increase their DVFS state and thus their computing power. In terms of accuracy, BPP and BSRWS-AR are therefore the most satisfying heuristics, but BPP is significantly faster than BSRWS-AR. This is mostly the case in our experiments.

Fig. 9 gives the runtimes of the MILP and the heuristics depending on power. Note that the y-axis is plotted on a logarithmic scale. There is a general trend for all the runtimes to increase with power. This is intuitive since the more power, the more machines the heuristics have to consider. Compared to the MILP that has an average runtime of 2.83 s per power value, the heuristics are more time efficient to find a configuration. The runtime of the BPP heuristic is of the order of milliseconds and increases slightly depending on power. While the BSRWS-AR heuristic runtime increases dramatically: from 0.1 ms to more than 1 s depending on power. This is partly due to the fact that the more machines are switched-on, the more configurations are explored. Note that the use of the *redistribute* function in BSRWS heuristic

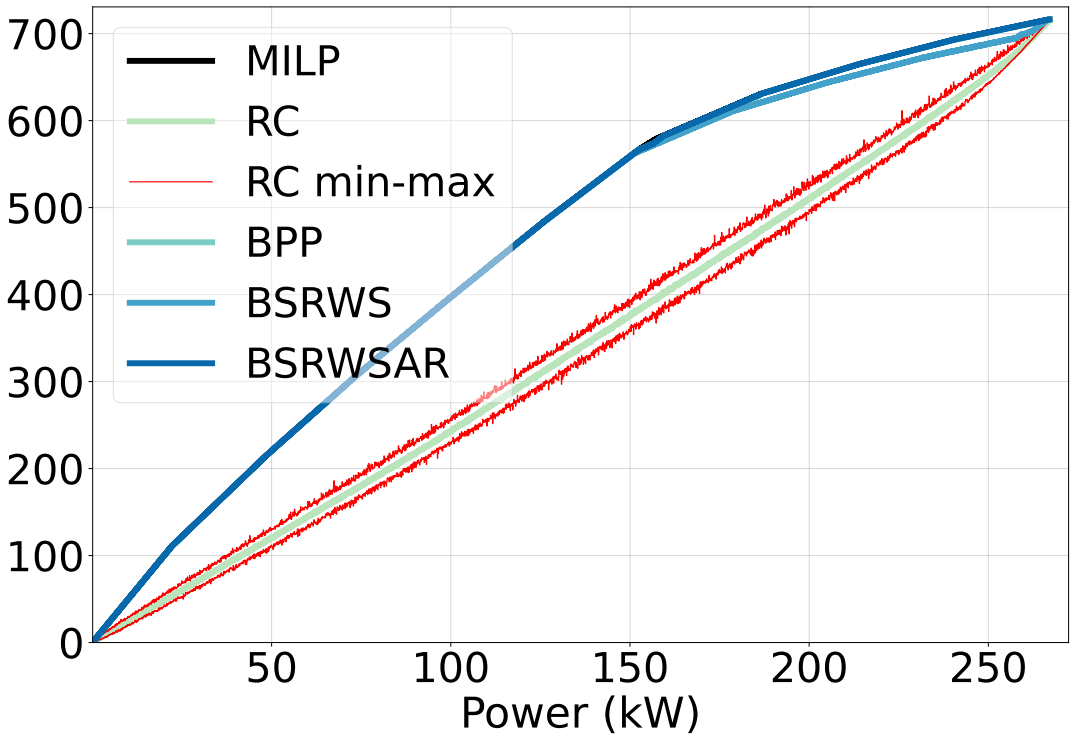


Figure 6: Comparison of the maximum computing power computed by the MILP and heuristics depending on power value from 63 W to 267 064 W by steps of 100 W

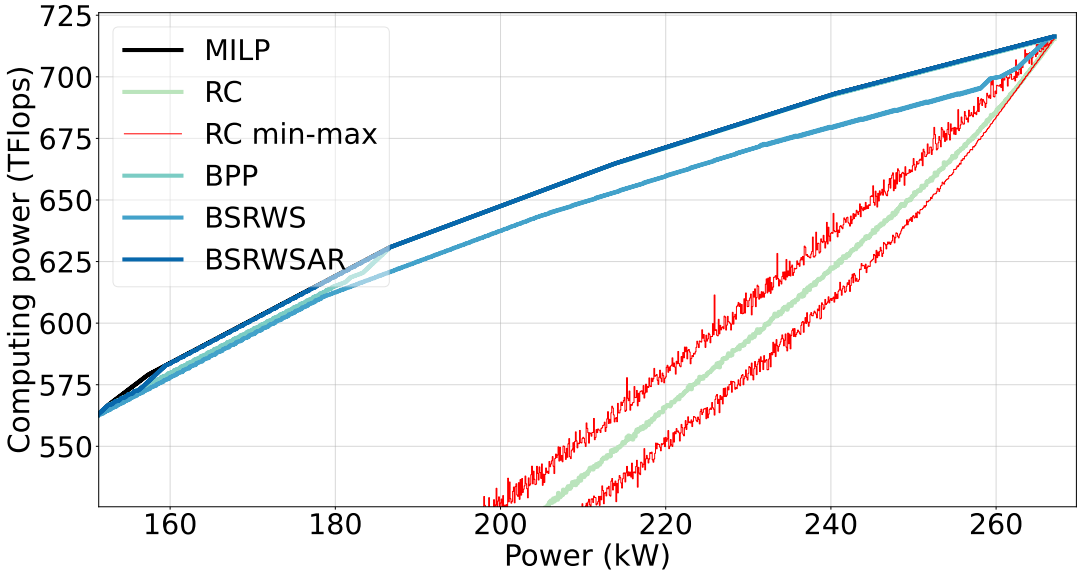


Figure 7: Comparison of the maximum computing power computed by the MILP and heuristics. Zoom between 140 kW and 267 kW from the Fig. 6

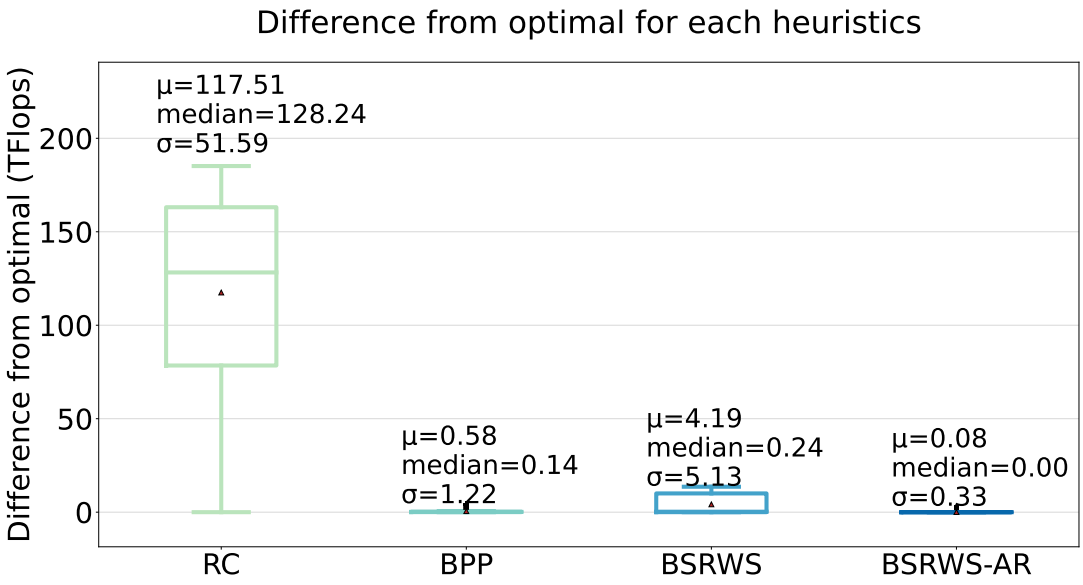


Figure 8: Difference from the optimal for each heuristics

Table 3: Average and median relative deviation in percentage of heuristics from optimal solution.

	MILP	RC	BPP	BSRWS	BSRWS-AR
Avg. dev. (%)	-	31.84	0.12	0.65	**0.03**
Median dev. (%)	-	34.57	0.04	0.09	**0.00**
Avg. Exec. Time (s)	2.83	1.15×10^{-3}	9.07×10^{-3}	$\mathbf{1.03 \times 10^{-3}}$	1.61
Min. Exec. Time (s)	0.94	7.31×10^{-6}	3.51×10^{-4}	3.46×10^{-5}	3.62×10^{-5}
Max. Exec. Time (s)	55.86	2.40×10^{-3}	1.23×10^{-2}	7.12×10^{-3}	4.09

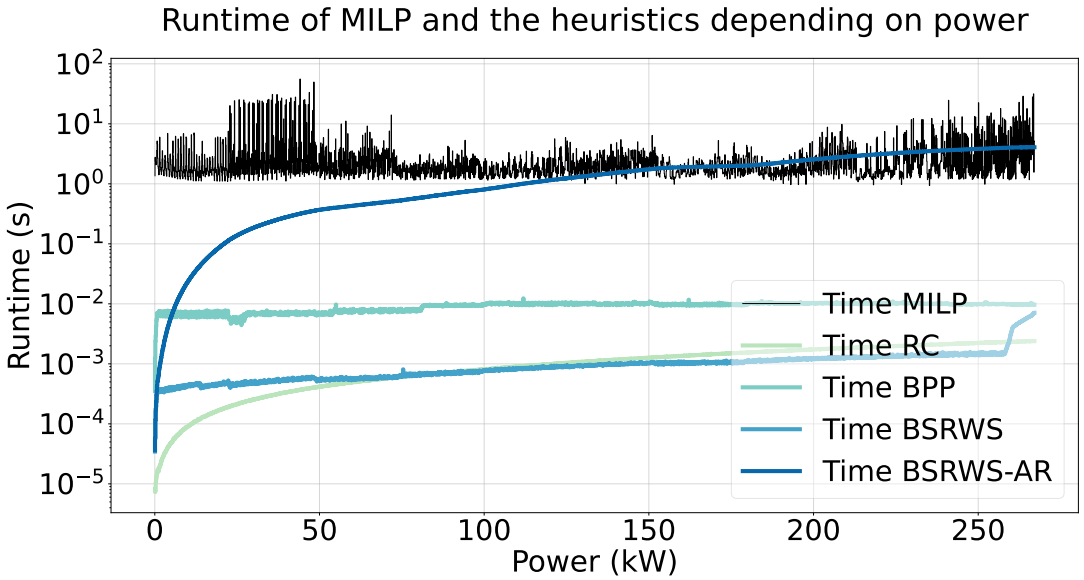


Figure 9: Runtime of the MILP and the heuristics depending on power

when all the machines are switched-on explains the increase of the runtime when the power is approximately 260 kW.

6 Conclusion

In this paper, we tackle the problem of minimizing a power value to switch-on just enough machines to process a workload over a time interval while respecting quality of service constraints. We propose a binary search algorithm to solve this problem with multiple variants. This algorithm uses two functions, one that computes the maximum computing power that can be obtained knowing a given power, and another that schedules the workload on the switched-on machines. Since computing the maximum processing power is suspected to be complex in the heterogeneous case, we propose a MILP and 3 non-trivial heuristics and compare their performance and runtime. Heuristics give satisfactory results in a reasonable time, with an average relative deviation from optimal solution of 0.12%, 0.65% and 0.03%. Looking at the results and runtime, the BPP (Balance Power-Performance) heuristic seems the most suitable to solve this problem in a reasonable time.

These different approaches show that using DVFS states in a heterogeneous environment allows approaching the optimal configuration of the machines and thus using energy efficiently. This is possible using the performance of the machines and their power consumption depending on DVFS states. These approaches contribute to the autonomy and reliability of the datacenter.

Several directions will be explored for future works. We will consider the switch-on and switch-off times of the machines and their respective power consumption in the schedule. This implies exploring the consequences of changing the configuration of the machines between two time intervals and analyzing the impact on power consumption and scheduling.

Acknowledgments

We would like to thank the ANR project ENERGUMEN (contract “ANR-18-CE25-0008”) and Grid5000 for providing some of the data used in this article.

References

[1] Energy saving in large scale distributed platforms – Energumen. `https://anr.fr/Project-ANR-18-CE25-0008`, 2018. [Online; accessed 20-November-2022].

[2] Apple: Environmental Progress Report (2022). `https://www.apple.com/environment/`, 2022. [Online; accessed 20-May-2022].

[3] Clusters Grid5000. `https://www.grid5000.fr/`, 2022. [Online; accessed 21-October-2022].

[4] M.T. Ahammed, N. Osman, C. Das, M.A. Hossain, S. Hossain, and M.H. Kaium. Analysis of energy consumption for a hybrid green data center. In *2022 International Conference on Innovations in Science, Engineering and Technology (ICISET)*, pages 318–323. IEEE, 2022.

[5] J. Arjona Aroca, A. Chatzipapas, A. Fernández Anta, and V. Mancuso. A measurement-based analysis of the energy consumption of data center servers. In *Proceedings of the 5th international conference on Future energy systems*, pages 63–74, 2014.

[6] F. Bordage. The environmental footprint of the digital world. *GreenIT*, page 20, 2019.

[7] P. Chaithra. Eco friendly green cloud computing. *Journal of Research Proceedings*, 1(2):41–52, 2021.

[8] K. Chang, S. Park, H. Kong, and W. Kim. Optimizing energy consumption for a performance-aware cloud data center in the public sector. *Sustainable Computing: Informatics and Systems*, 20:34–45, 2018.

[9] T. Ciesielczyk, A. Cabrera, A. Oleksiak, W. Piątek, G. Waligóra, F. Almeida, and V. Blanco. An approach to reduce energy consumption and performance losses on heterogeneous servers using power capping. *Journal of Scheduling*, 24:489–505, 2021.

[10] Q. Fang, J. Wang, Q. Gong, and M. Song. Thermal-aware energy management of an hpc data center via two-time-scale control. *IEEE Transactions on Industrial Informatics*, 13(5):2260–2269, 2017.

[11] Greendatanet research project. `http://www.greendatanet-project.eu/`, 2013–2016. Accessed: 2021-05-28.

[12] Y.F. Hsu, K. Matsuda, and M. Matsuoka. Self-aware workload forecasting in data center power prediction. In *2018 18th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, pages 321–330. IEEE, 2018.

[13] H.A. Kurdi, S.M. Alismail, and M.M. Hassan. Lace: A locust-inspired scheduling algorithm to reduce energy consumption in cloud datacenters. *IEEE Access*, 6:35435–35448, 2018.

[14] W.Y. Lee. Energy-efficient scheduling of periodic real-time tasks on lightly loaded multicore processors. *IEEE Transactions on Parallel and Distributed Systems*, 23(3):530–537, 2011.

[15] X. Li, P. Garraghan, X. Jiang, Z. Wu, and J. Xu. Holistic virtual machine scheduling in cloud datacenters towards minimizing total energy. *IEEE Transactions on Parallel and Distributed Systems*, 29(6):1317–1331, 2017.

[16] H. Liu, B. Liu, L.T. Yang, M. Lin, Y. Deng, K. Bilal, and S.U. Khan. Thermal-aware and dvfs-enabled big data task scheduling for data centers. *IEEE Transactions on Big Data*, 4(2):177–190, 2017.

[17] E. Masanet, A. Shehabi, N. Lei, S. Smith, and J. Koomey. Recalibrating global data center energy-use estimates. *Science*, 367(6481):984–986, 2020.

[18] M. Masdari and A. Khoshnevis. A survey and classification of the workload forecasting methods in cloud computing. *Cluster Computing*, 23(4):2399–2424, 2020.

[19] S. Mazumdar and M. Pranzo. Power efficient server consolidation for cloud data center. *Future Generation Computer Systems*, 70:4–16, 2017.

[20] B. Nikzad, B. Barzegar, and H. Motameni. Sla-aware and energy-efficient virtual machine placement and consolidation in heterogeneous dvfs enabled cloud datacenter. *IEEE Access*, 10:81787–81804, 2022.

[21] A. Pahlevan, M. Rossi, P. Garcia del Valle, D. Brunelli, and D. Atienza Alonso. Joint computing and electric systems optimization for green datacenters. Technical report, Springer, 2017.

[22] K. Pedretti, R.E. Grant, J.H. Laros III, M. Levenhagen, S.L. Olivier, L. Ward, and A.J. Younge. A comparison of power management mechanisms: P-states vs. node-level power cap control. In *International Parallel and Distributed Processing Symposium Workshops*. IEEE, 2018.

[23] Y. Peng, D.K. Kang, F. Al-Hazemi, and C.H. Youn. Energy and qos aware resource allocation for heterogeneous sustainable cloud datacenters. *Optical Switching and Networking*, 23:225–240, 2017.

[24] J.M. Pierson, G. Baudic, S. Caux, B. Celik, G. Da Costa, L. Grange, M. Haddad, J. Lecuivre, J.M. Nicod, L. Philippe, V. Rehn-Sonigo, R. Roche, G. Rostirolla, A. Sayah, P. Stolf, M.T. Thi, and C. Varnier. Datazero: Datacenter with zero emission and robust management using renewable energy. *IEEE Access*, 7:103209–103230, 2019.

[25] C. Reiss, A. Tumanov, G.R. Ganger, R.H. Katz, and M.A. Kozuch. Heterogeneity and dynamicity of clouds at scale: Google trace analysis. In *Proceedings of the third ACM symposium on cloud computing*, pages 1–13, 2012.

[26] C. Reiss, J. Wilkes, and J. L Hellerstein. Google cluster-usage traces: format+ schema. *Google Inc., White Paper*, 2011.

[27] B. Sharma, V. Chudnovsky, J.L. Hellerstein, R. Rifaat, and C.R. Das. Modeling and synthesizing task placement constraints in google compute clusters. In *Proceedings of the 2nd ACM Symposium on Cloud Computing*, pages 1–14, 2011.

[28] Standard performance evaluation corporation. `http://spec.org/`. Accessed: 2023-02-28.

[29] M.T. Thi, J.M. Pierson, G. da Costa, P. Stolf, J.M. Nicod, G. Rostirolla, and M. Haddad. Negotiation Game for Joint IT and Energy Management in Green Datacenters. *Future Generation Computer Systems*, 110:1116–1138, 2020.

[30] M.T. Thi, J.M. Pierson, G. Da Costa, P. Stolf, J.M. Nicod, G. Rostirolla, and M. Haddad. Negotiation game for joint it and energy management in green datacenters. *Future Generation Computer Systems*, 110:1116–1138, 2020.

[31] W. Wang, B. Li, and B. Liang. Dominant resource fairness in cloud computing systems with heterogeneous servers. In *IEEE INFOCOM 2014-IEEE Conference on Computer Communications*, pages 583–591. IEEE, 2014.

Table 4: Average and median relative deviation in percentage of heuristics from the optimal in the homogeneous case with 300 machines for one type of machine (Taurus).

	RC	BPP	BSRWS	BSRWS-AR
Avg. dev. (%)	0.22	0.22	0.22	0.22
Median dev. (%)	0.07	0.07	0.07	0.07

[32] R. Yadav, W. Zhang, K. Li, C. Liu, M. Shafiq, and N.K. Karn. An adaptive heuristic for managing energy consumption and overloaded hosts in a cloud data center. *Wireless Networks*, 26:1905–1919, 2020.

[33] H. Zhang and H. Hoffmann. Podd: Power-capping dependent distributed applications. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–23, 2019.

[34] M. Zhao, X. Wang, and J. Mo. Workload and energy management of geo-distributed datacenters considering demand response programs. *Sustainable Energy Technologies and Assessments*, 55:102851, 2023.

A Other experiments

A.1 Homogeneous case

Figure 10 shows the results of one of our experiments in the homogeneous case with 300 machines for one type of machine (Taurus). Figure 10a shows the best ratio of the machine depending on the power, all DVFS states considered. Figure 10b shows the solutions proposed by the MILP and the different heuristics depending on power. In the homogeneous case, we observe a significant linearity between the computing power that can be delivered and the power. This is why the RC heuristic performs well. In this case, the heuristics have equivalent results, as shown in the Figures 10c and 10d and in Table 4.

A.2 Heterogeneous case

Figure 11, 12 and 13 show the results of some of our experiments in the heterogeneous case with 200 machines for 2 types of machines, 100 machines of each type. Figure 11b, 12b and 13b show the solutions proposed by the MILP and the different heuristics depending on power. In the heterogeneous case, the computing power that can be delivered can be linear with power. This occurs when the ratios of the machines are equivalent. In the case where the ratios are different, the linearity is broken and the RC heuristic fails to perform well as shown in the Figure 11c. The other heuristics provide excellent results in all cases as shown in Table 5, 6 and 7. The BPP heuristic can outperform the others on some experiments and under-perform BSRWS and BSRWS-AR on others. Also, it is possible that BSRWS-AR fails to improve the results of BSRWS. The same conclusions can be made about the execution times of the heuristics as in the homogeneous case. However, in some cases, it is observed that the execution time of BSRWS-AR explodes and reaches the execution time of MILP when the input power is high.

Figure 14 shows the results of one of our experiments in the heterogeneous case with 500 machines for 5 types of machines, 100 machines of each type. The same Figure 14b show the solutions proposed by the MILP and the different heuristics depending on power. Except for RC, the heuristics provide excellent results. In this case, the BPP heuristic outperform BSRWS

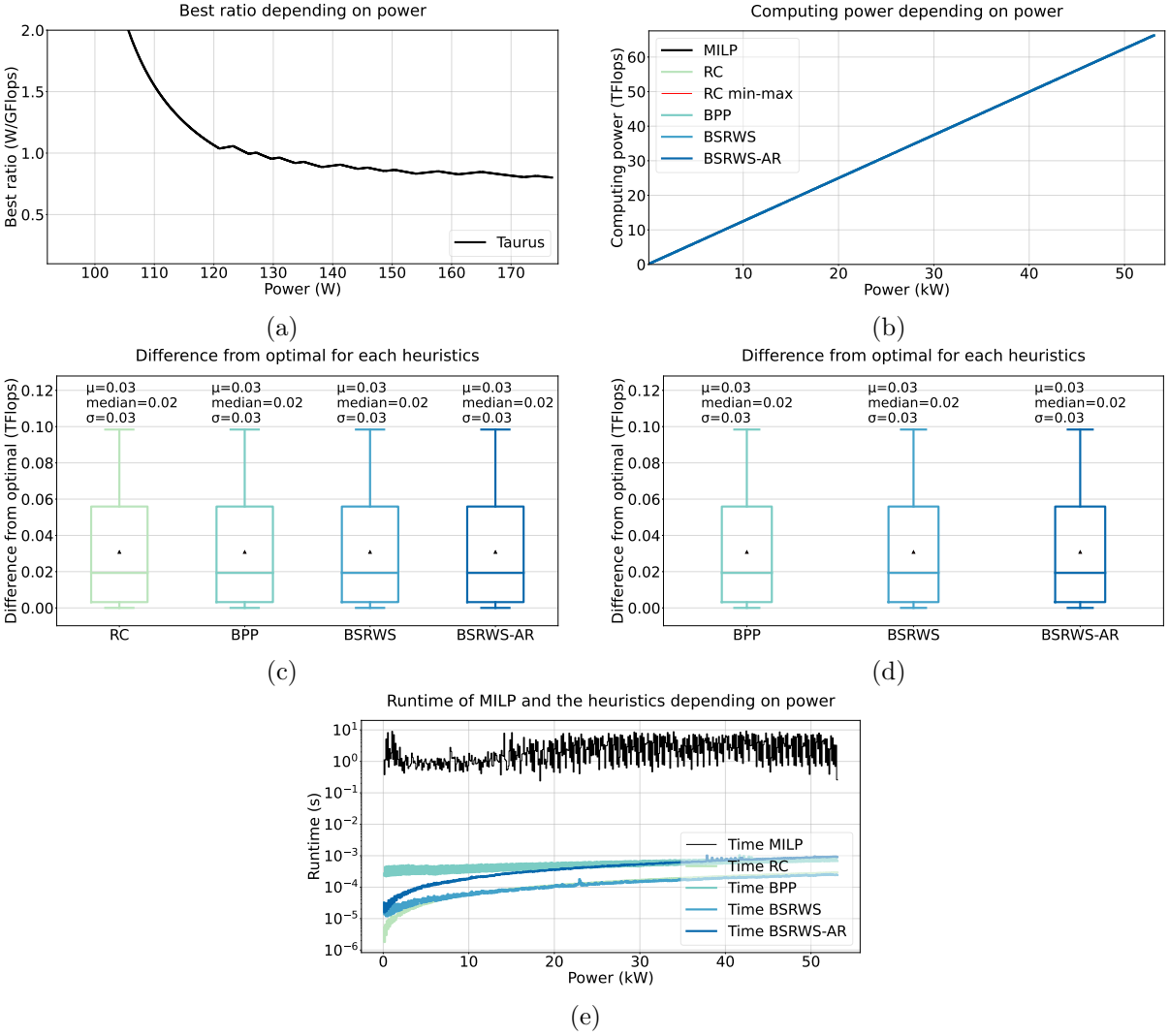


Figure 10: An experiment in the homogeneous case with 300 machines for one type of machine (Taurus).

Table 5: Average and median relative deviation in percentage of heuristics from the optimal in the heterogeneous case with 200 machines for 2 types of machines, 100 machines of each type (Taurus & Parasilo).

	RC	BPP	BSRWS	BSRWS-AR
Avg. dev. (%)	25.40	0.02	0.02	0.02
Median dev. (%)	32.43	0.00	0.00	0.00

Table 6: Average and median relative deviation in percentage of heuristics from the optimal in the heterogeneous case with 200 machines for 2 types of machines, 100 machines of each type (Grisou & Grimoire).

	RC	BPP	BSRWS	BSRWS-AR
Avg. dev. (%)	4.07	0.28	0.40	0.27
Median dev. (%)	4.50	0.19	0.33	0.26

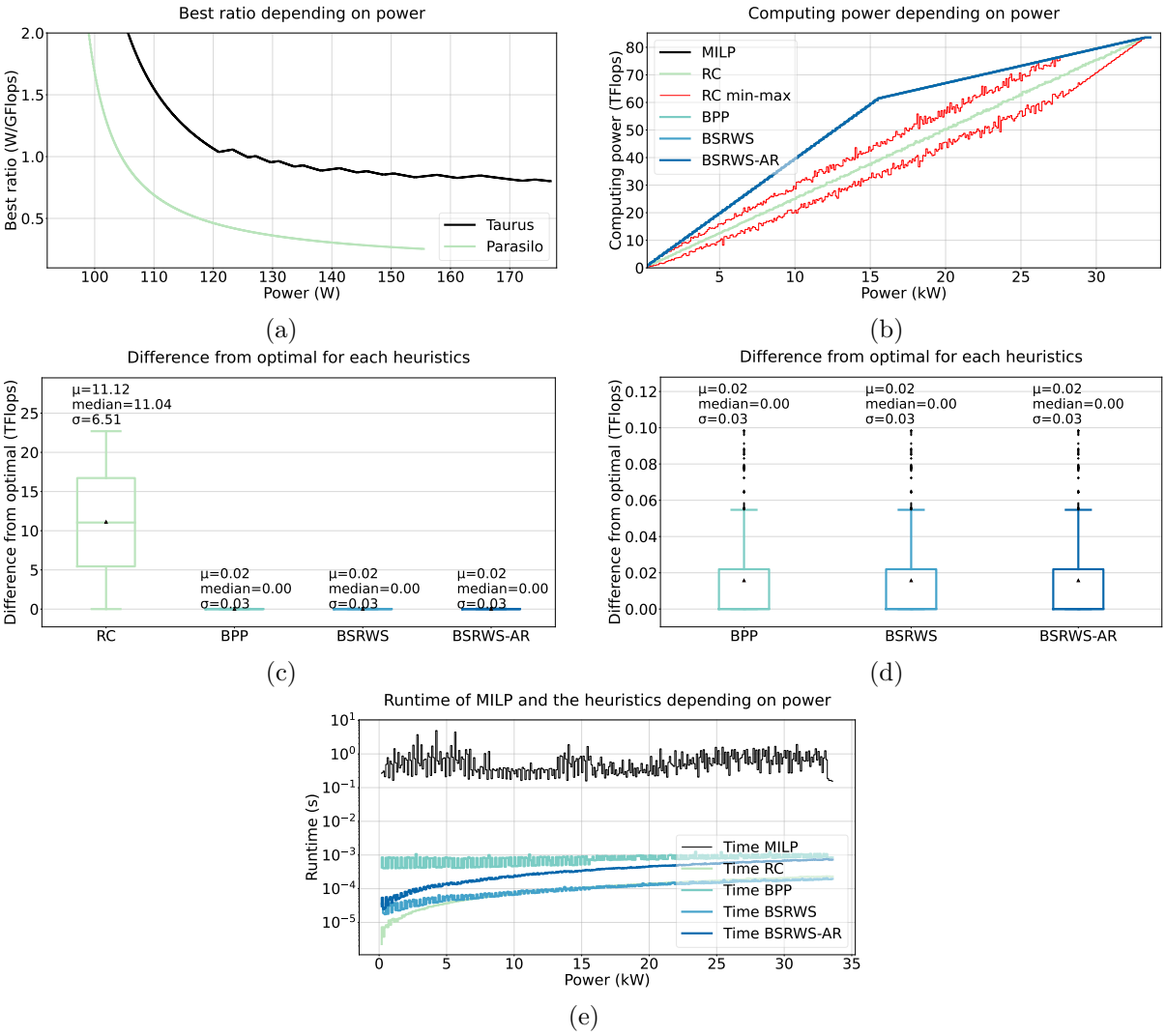


Figure 11: An experiment in the heterogeneous case with 200 machines for 2 types of machines, 100 machines of each type (Taurus & Parasilo).

Table 7: Average and median relative deviation in percentage of heuristics from the optimal in the heterogeneous case with 200 machines for 2 types of machines, 100 machines of each type (Chifflet & Grele).

	RC	BPP	BSRWS	BSRWS-AR
Avg. dev. (%)	4.38	0.34	0.14	0.06
Median dev. (%)	4.98	0.21	0.00	0.00

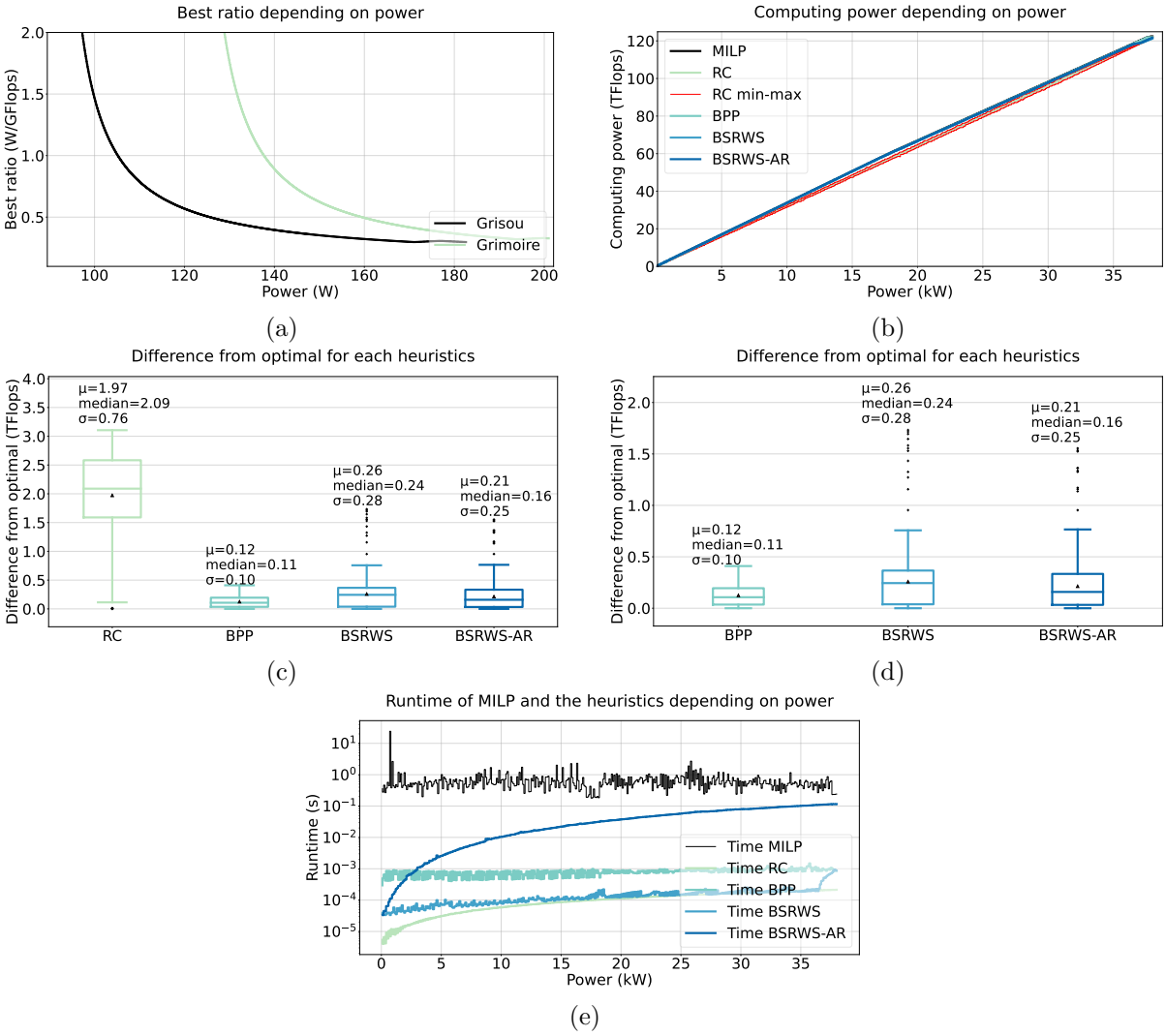


Figure 12: An experiment in the heterogeneous case with 200 machines for 2 types of machines, 100 machines of each type (Grisou & Grimoire).

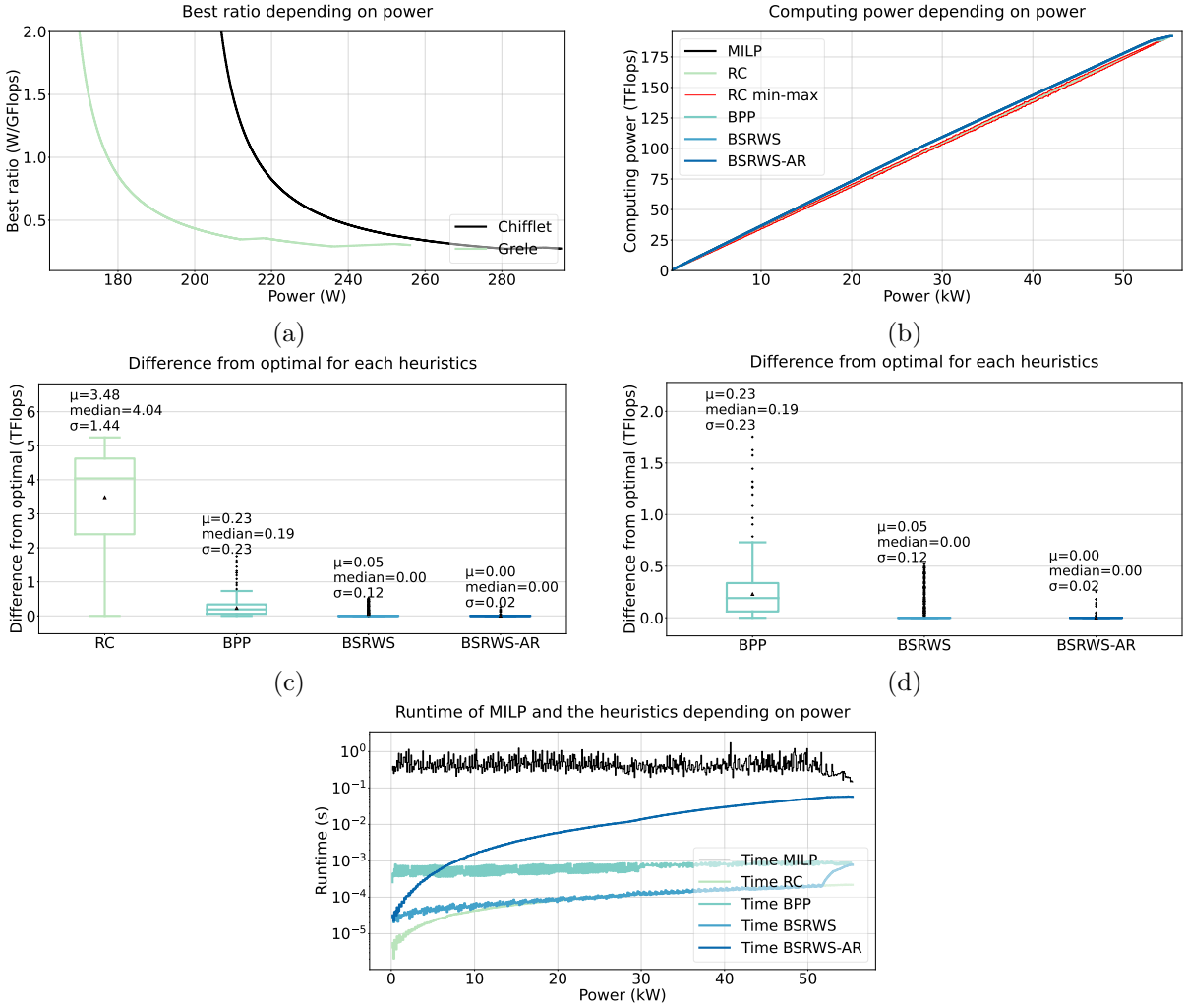


Figure 13: An experiment in the heterogeneous case with 200 machines for 2 types of machines, 100 machines of each type (Chifflet & Grele).

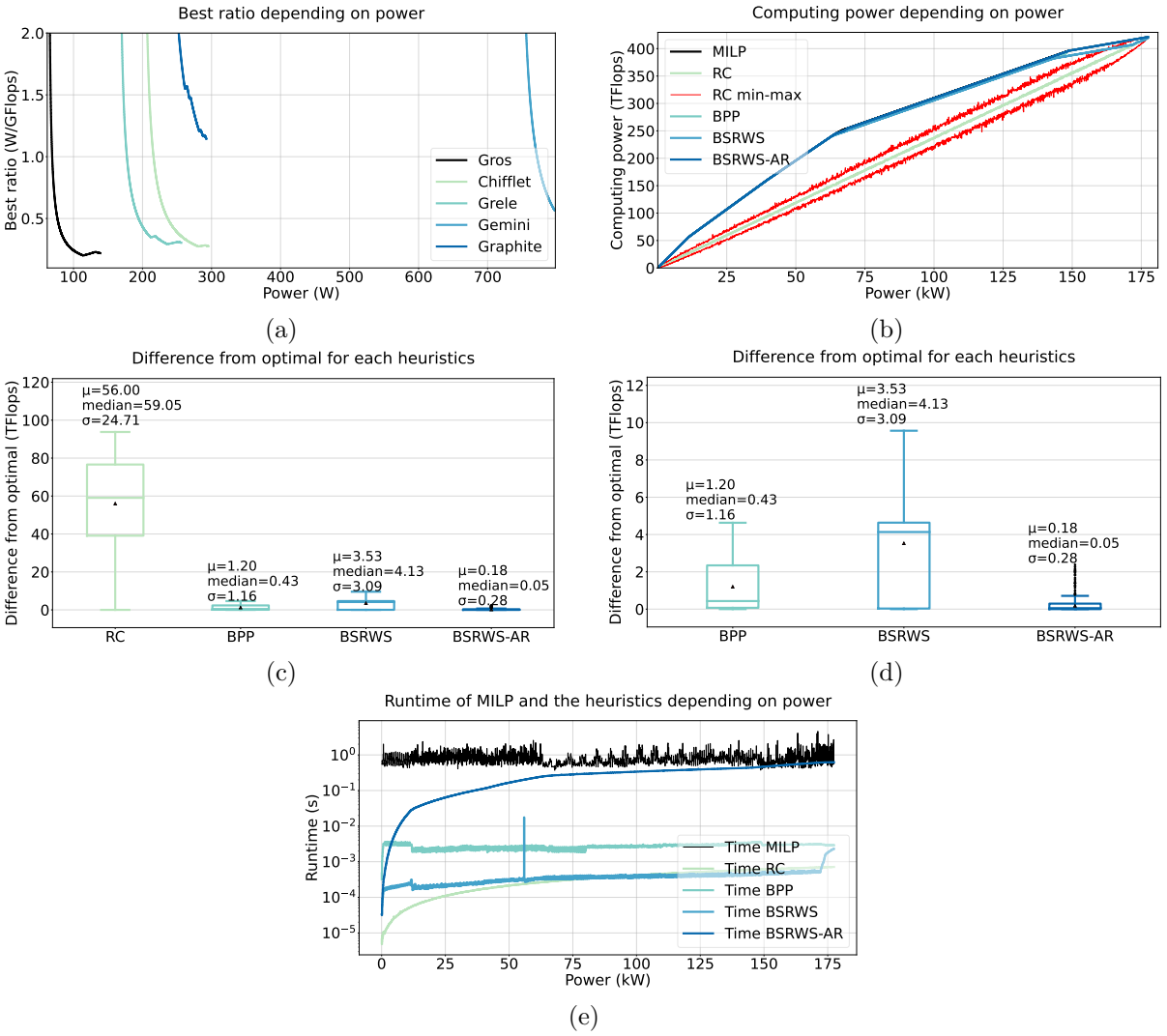


Figure 14: An experiment in the heterogeneous case with 500 machines for 5 types of machines, 100 machines of each type (Gros, Chifflet, Grele, Gemini & Graphite).

Table 8: Average and median relative deviation in percentage of heuristics from the optimal in the heterogeneous case with 500 machines for 5 types of machines, 100 machines of each type (Gros, Chifflet, Grele, Gemini & Graphite).

	RC	BPP	BSRWS	BSRWS-AR
Avg. dev. (%)	27.41	0.42	1.03	0.07
Median dev. (%)	27.47	0.33	1.24	0.02

and underperform BSRWS-AR as it successfully improved the results of BSRWS, as shown in Table 8.

Table 9 shows the percentage of experiments where the BSRWS-AR heuristic improves the BSRWS heuristic results depending on the number of machines and the number of types of machines (10 experiments per cell). The results show that the more heterogeneous the datacenter, the more the BSRWS-AR heuristic improves the results of the BSRWS heuristic. However, the number of machines in the datacenter does not seem to have an impact on the

Table 9: percentage of experiments where the BSRWS-AR heuristic improves the BSRWS heuristic results depending on the number of machines and the number of types of machines (10 experiments per cell).

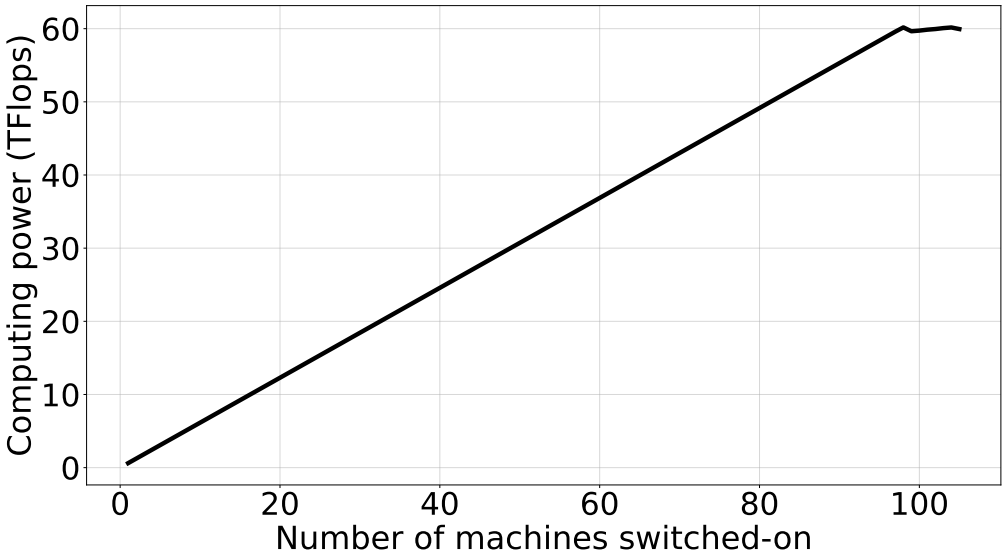
	1 type	2 types	3 types	4 types	5 types	6 types
60 machines	50%	70%	80%	80%	80%	90%
120 machines	50%	70%	80%	80%	80%	90%
180 machines	50%	70%	80%	80%	80%	90%
240 machines	50%	70%	80%	80%	80%	90%
300 machines	50%	70%	80%	80%	80%	90%

Table 10: percentage of machine configurations that are sufficient to explore to reach the best BSRWS-AR solution depending on the number of machines and the number of types of machines (10 experiments per cell).

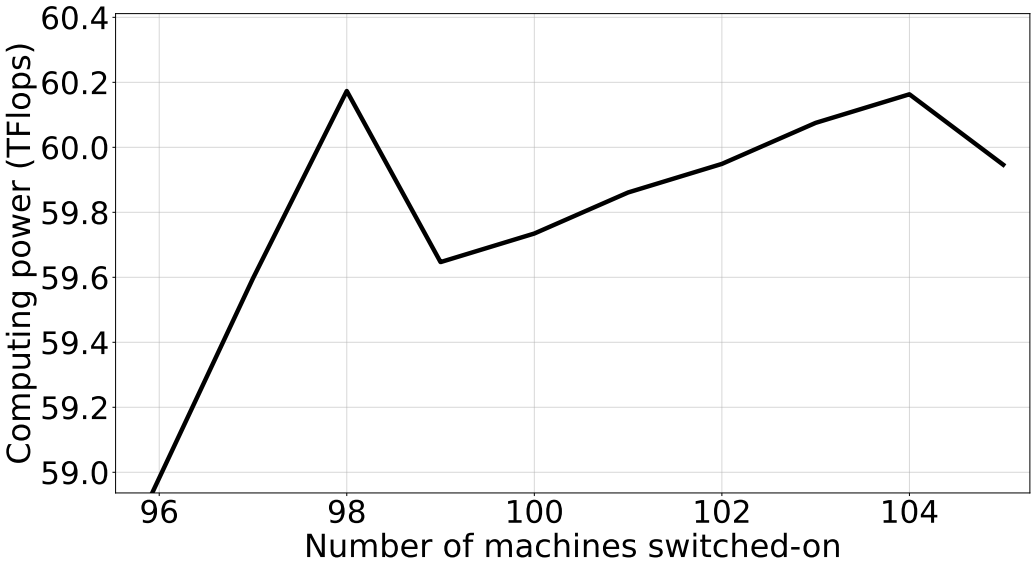
	1 type	2 types	3 types	4 types	5 types	6 types
60 machines	20%	20%	20%	16.67%	16.67%	12.5%
120 machines	20%	20%	20%	16.67%	16.67%	12.5%
180 machines	20%	20%	20%	16.67%	16.67%	12.5%
240 machines	20%	20%	20%	16.67%	16.67%	12.5%
300 machines	20%	20%	20%	16.67%	16.67%	12.5%

improvement of the results. the choice of BSRWS-AR seems more judicious than BSRWS when the datacenter is heterogeneous

The more machines the BSRWS heuristic switches-on, the more configurations BSRWS-AR explores, *i.e.* as many configurations as machines switched-on by BSRWS. However, it is not necessary to explore all configurations. For example, it is not necessary to explore configurations with 1, 10, 25 or even 50 machines if the solution proposed by BSRWS, which is close to the optimal solution, is a configuration with more or less 100 switched-on machines, as shown in the Figures 15 and 16. It is therefore possible to reduce the execution time of BSRWS-AR by reducing the number of configurations explored. To determine the maximum number of configurations to explore, several experiments were executed in the homogeneous and heterogeneous case with different number of machines and different number of types of machines. Table 10 shows the percentage of machine configurations that are sufficient to explore to reach the best BSRWS-AR solution depending on the number of machines and the number of types of machines (10 experiments per cell). The more heterogeneous the datacenter is, the more it is possible to reduce the number of configurations explored, and thus to reduce the execution time of BSRWS-AR. The number of machines has no influence.

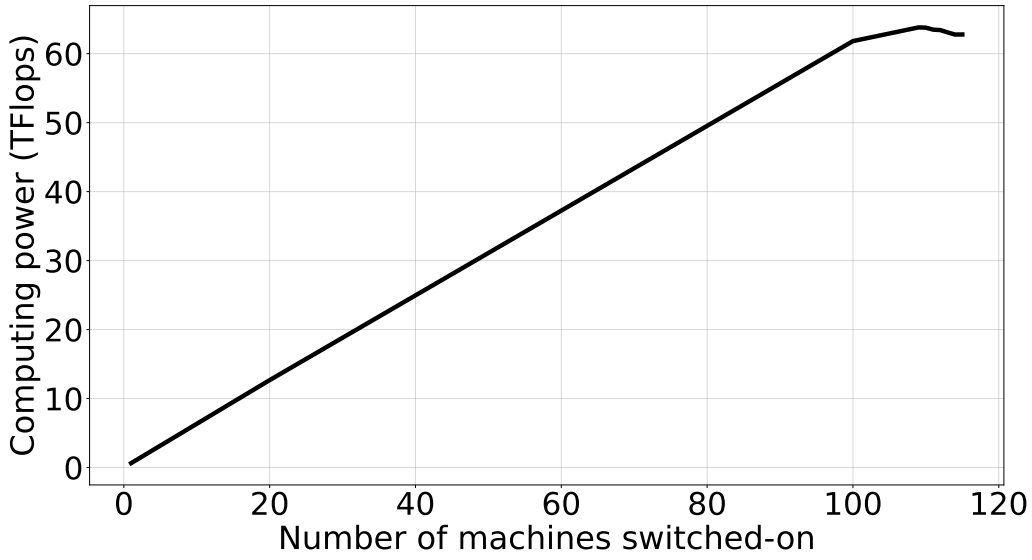

Computing power (TFlops)
0
10
20
30
40
50
60
0
20
40
60
80
100
Number of machines switched-on

(a)

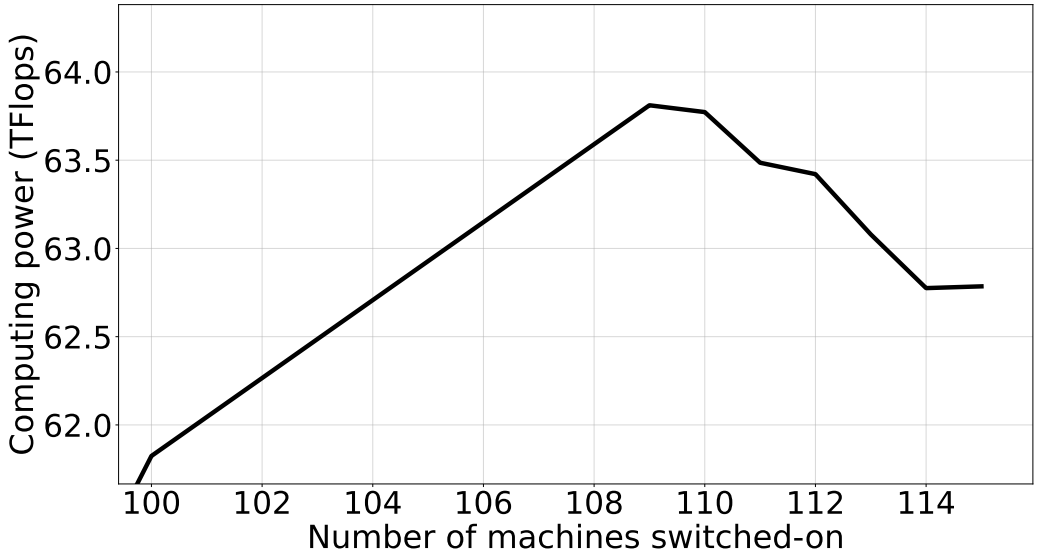

Computing power (TFlops)
59.0
59.2
59.4
59.6
59.8
60.0
60.2
60.4
96
98
100
102
104
Number of machines switched-on

(b)

Figure 15


Computing power (TFlops)
0
10
20
30
40
50
60
0
20
40
60
80
100
120
Number of machines switched-on

(a)


Computing power (TFlops)
62.0
62.5
63.0
63.5
64.0
100
102
104
106
108
110
112
114
Number of machines switched-on

(b)

Figure 16

FEMTO-ST INSTITUTE, headquarters
15B Avenue des Montboucons - F-25030 Besançon Cedex France
Tel: (33 3) 63 08 24 00 – e-mail: contact@femto-st.fr

FEMTO-ST — AS2M: TEMIS, 24 rue Alain Savary, F-25000 Besançon France
FEMTO-ST — DISC: UFR Sciences - Route de Gray - F-25030 Besançon cedex France
FEMTO-ST — ENERGIE: Parc Technologique, 2 Av. Jean Moulin, Rue des entrepreneurs, F-90000 Belfort France
FEMTO-ST — MEC'APPLI: 24, chemin de l'épitaphe - F-25000 Besançon France
FEMTO-ST — MN2S: 15B Avenue des Montboucons - F-25030 Besançon cedex France
FEMTO-ST — OPTIQUE: 15B Avenue des Montboucons - F-25030 Besançon cedex France
FEMTO-ST — TEMPS-FREQUENCE: 26, Chemin de l'Epitaphe - F-25030 Besançon cedex France

http://www.femto-st.fr