Empirical Comparison of Power-efficient Virtual Machine Assignment Algorithms

Jordi Arjona Aroca
Nokia Bell Labs, Dublin, Ireland
Email: jorge.arjona_aroca@nokia.com

Antonio Fernández Anta
IMDEA Networks Institute, Madrid, Spain
Email: antonio.fernandez@imdea.org

***Abstract*—The advent of cloud computing has changed the way many companies do computation, allowing them to outsource it to the cloud. This has given origin to a new kind of business, the cloud providers, which run large datacenters. In order to be competitive, cloud providers must keep their operational costs low. One way to reduce these costs is reducing the energy consumed with smart task assignment algorithms, which decide where tasks are to be placed upon their arrival. Unfortunately, almost no task assignment algorithm used is power aware.**

In this paper we compare the performance of multiple task assignment algorithms for saving energy. We assume that tasks are in fact virtual machines that have to be assigned to physical machines, and we assume that the physical machines have a power consumption that increases superlinearly with the load. First, we propose two tunable power-aware task assignment algorithms (that subsume the algorithms studied in [15]). These algorithms are then compared with multiple state-of-the-art algorithms in different meaningful scenarios. Both algorithms prove themselves as interesting assignment algorithms since, properly configured, they outperform the other algorithms in most of the cases.

***Index Terms*—Cloud computing, Datacenters, Virtual Machine Assignment, Energy Efficiency, Scheduling, Load Balancing.**

I. Introduction

Cloud computing has brought a huge number of new possibilities to the business world. For instance, it allows companies to outsource large amounts of computation to servers in the cloud [12], [6], [10], [3]. Outsourcing computation to the cloud allows companies to reduce their capital expenditures (CAPEX) eliminating the need of acquiring computers and network equipment that would be underused and soon outdated. On the other hand, not owning this hardware, which, used or not, is usually kept running 24/7 implies also a reduction of their operational expenditure (OPEX). The cloud allows companies to adapt computational resources to their business needs, thus making these costs variable. The fixed CAPEX an OPEX have been shifted to the cloud providers, such as Amazon Web Services [2], Microsoft Azure [6], Rackspace [10], or Citrix [3], which can offer a similar service at a lower cost because they run huge datacenters (benefiting of an economy of scale) and leverage the fact that demands are stochastic.

Although the profits of cloud providers are in the order of billions of dollars [1], [4], [11], they could enlarge their margins by reducing the energy they consume [12] as well as increasing the efficiency of their servers as much as possible. Virtualization plays a key role in the efficient use of servers, allowing the sharing of computational resources by multiple customers via Virtual Machines (VM)[1]. However, virtualization is like a play with multiple actors, and one of the leading actors in this play is the task assignment algorithm. The resources used by a VM are in fact resources of the Physical Machines (PM) in which it resides (it runs). Assignment (or allocation) algorithms are in charge of deciding how this VM to PM assignment is done upon VMs arrival. Traditionally, bin-packing-like algorithms, *i.e.* algorithms that attempt to minimize the number of PMs required to host the tasks arriving into the system, has been considered to be the best possible solution. The belief was that using less PMs increases the overall efficiency. Hence the most common state of the art allocation algorithms follow this approach: *First Fit*, *Packing* or *Most Full First*[2].

However, trying to concentrate all the VMs in the smallest possible number of PMs is not the best way to be energy efficient, unlike it has been usually assumed. This assumption was due to the fact that, in the past, PMs used to be single core, and not able to change their frequency of operation at wish. This led to a linear dependency between the power consumption and the load, usually assumed in many works in the literature [17], [34], [28], [23]. However, after the arrival of multicore machines the rules have changed. As shown in [13], nowadays, the power consumption of data center servers has a superlinear dependency on their load. This non-linear dependency implies the existence of optimal loads and frequencies and paying higher costs, in terms of power consumption, when PMs are loaded beyond that combination of optimal load and frequency. Thus, according to the results that can be observed in nowadays servers [13], due to this superlinear dependency on the load, loading servers up to an optimal load, even when it implies using a few more servers than with a bin-packing algorithm, can save a considerable amount of energy. Therefore, two interesting issues arise: (a) Dynamic Voltage and Frequency Scaling (DVFS) may not be enough to manipulate the frequencies of operation of servers and reduce servers energy consumption, as noted in [14] and [21]; (b) the minimization of the number of PMs might not be

[1]We use task or VM indistinctly, assuming that each task that arrives to the system is run in a different VM.

[2]These algorithms are described in Section V.

978-3-901882-70-8

the optimal solution anymore, leading to the need of rethinking how VMs are assigned when consolidation is applied. A theoretical study of this problem, a.k.a. the Virtual Machine Assignment Problem (VMA), assuming a superlinear power model is presented in [15], where a competitive analysis of multiple offline and online cases of this problem is provided.

This document complements the theoretical work presented in [15] and extends the empirical analysis performed in [14]. New assignment algorithms are presented and compared, via simulation, to our own devised previous online algorithms, improving the results presented in [14], as well as to multiple state-of-the-art assignment algorithms currently used in popular cloud computing platforms such as Openstack [8], OpenNebula [29], Koala [16], or Eucalyptus [22]. As mentioned, the algorithms proposed for cloud systems rarely take power consumption into account, hence the interest of power-aware new algorithms. With respect to the power-aware algorithms studied in [14] (that include the algorithm of [15]), we generalize these algorithms, introducing a parameter β, which is a threshold on the load allocated to a physical machine, and hence algorithm β-*VMA* (where the acronym *VMA* comes from *Virtual Machine Assignment Algorithm*). Then, the algorithms *VMA1* and *VMA2* proposed in [15] and [14] are now only the particular cases of β-*VMA* where $\beta = \{1, 2\}$. We also perform a study of the effect of the new β parameter in the performance. In addition, we propose a new assignment algorithm, φ-*VMA*, that takes into account the current rate at which a physical machine is working and the rate at which it would work if a load was assigned to it. Hence, the main contributions of this paper are three. First, we propose β-*VMA*, that generalizes our previously proposed algorithms VMA1 and VMA2, and study the impact of varying β. Second, we present φ-*VMA*, a novel power aware assignment algorithm. Last but not least, we make a detailed study of two versions of the VMA problem (described in [15]), and compare the performance of the generalized VMA algorithm, β-*VMA*, the newly proposed φ-*VMA*, and multiple state-of-the-art algorithms in different meaningful scenarios. Both β-*VMA* and φ-*VMA* prove themselves as valid assignment algorithms, since they outperform the other algorithms in most of the cases.

Roadmap. The paper is organized as follows. An overview related work is presented in Section II. A brief introduction to the VMA problem, and to the cases that we analyze, is done in Section III. Section IV introduces φ-*VMA* and β-*VMA*. The empirical comparison between our algorithms and the state of the art ones is described and presented in Section V. Section VI closes the paper with our conclusions and future work.

II. Related Work

The empirical and theoretical work related to VMA is vast and its detailed overview is out of the scope of this paper. We focus on empirical references and refer the reader to [15], and the references therein, to find some theoretical context.

Srikantaiah *et al.* [32] showed that VMA cannot be reduced to a bin packing problem if our goal is an energy efficient assignment. That is, assuming linear energy models and trying to minimize the number of PMs used is not the optimal solution. Therefore, energy models such as the proposed in [15], where the optimal load of a PM is a function only of the fixed cost of being active (b) and the exponential rate of power increase on the load (α) (that is, the optimal load is not related to the maximum capacity of a PM) seem to be more appropriate. Based on this energy model, we compare, in this work, some of the allocation policies presented in [15], [25] and [27].

Jansen and Brenner [25] evaluate the allocation of VMs to clusters following 9 placement policies, some of them included in popular cloud platforms like OpenNebula [29] or Eucalyptus [22], considering a linear energy cost model. The policies they consider are, namely, Round Robin, Striping, Packing, Load Balancing (free CPU count), Load Balancing (free CPU ratio), Watts per Core, and Cost per Core. We will adapt 5 of these policies to our model and cost function for the purpose of simulations.

Mills *et al.* [27] provide an objective method to compare different placement algorithms. They perform their analysis using 3 different cluster choice criteria and 6 different allocation heuristics from the bin packing literature, that are used in cloud platforms such as Koala [16] or Eucalyptus [22]. These policies are, namely, First Fit, Least-Full First, Most Loaded First, Next Fit, Random Assignment, and Tag & Pack.

In this work we compare our algorithms against most of this placement algorithms as we consider that they are still valid, being the scheduling policies used in the latest versions of popular cloud platforms like Eucalyptus [5] or OpenNebula [7]. Openstack, one of the most popular cloud platforms nowadays, does not exactly follow any of these policies but a filter scheduling approach [9]. This approach is, though, similar to the bin packing solutions as it first applies a set of filters to determine which is the set of valid PMs where a new virtual machine can be placed. Once these PMs have been determined, the one consuming the least resources is, by default, the chosen PM, resembling, then, a Least-Full First algorithm.

It is also worth to note that some works in the literature, like [17], [18], propose solutions that involve the initial placement of VMs as well as migration like. This migration is a consequence of considering arrival and termination of tasks. Upon the termination of a task, the optimal mapping of VMs to PMs may change and migration might be required to reconfigure the system in order to reach this new optimal placement. Of course this reconfiguration comes at a cost. Furthermore, placement algorithms complexity can be extended by considering some other variables like data center PUE [26] or traffic between VMs [33]. However, our study aims at a more fundamental problem, the optimal load at which a server run according to the way it consumes power. Therefore, all these aspects are not captured by our model although some have been left as future work. Therefore comparing these works with the placement algorithms we proposed is out of scope of this paper.

Finally, this work complements [15] and extends [15]. In

[15] Arjona *et al.* studied the VMA problem, performing a thorough offline and online analysis of multiple of its subproblems. In particular, they consider four different cases defined by whether the number of PMs or their capacity are bounded or unbounded, namely $(\cdot,\cdot)$-VMA, $(C,\cdot)$-VMA, $(\cdot,m)$-VMA and (C,m)-VMA. The main novelty on their study, inherited by this work, is a superlinear energy consumption model and an online allocation algorithm, denoted as *VMA1*. *VMA1* is the particular case $\beta = 1$ of the β-*VMA* algorithm proposed in this study.

The work presented in [15] is extended by proposing 2 new algorithms that improve the results obtained by *VMA1* and *VMA2*. The first of this algorithms, β-*VMA*, is a generalization of the previously proposed *VMA1* and *VMA2*. The second one, φ-*VMA*, that performs the assignment of VMs based on the current rate of the PMs in the cluster. Finally, note that the main difference between most of the algorithms in [25] and [27] and β-*VMA* and φ-*VMA* is that the latter provide power aware assignments while the former do not.

III. The VMA Problem

The VMA problem is described in [15] as follows. Given a set $S = \{s_1, \ldots, s_m\}$ of $m > 1$ identical physical machines (PMs) of capacity C; rational numbers μ, α and b, where $\mu > 0$, $\alpha > 1$ and $b > 0$; a set $D = \{d_1, \ldots, d_n\}$ of n virtual machines and a function $\ell : D \to \mathbb{R}$ that gives the CPU load each virtual machine incurs, the aim is to obtain a partition $\pi = \{A_1, \ldots, A_m\}$ of D, such that $\ell(A_i) \leq C$, for all i. Then, the objective is minimizing the power consumption of the set S of m PMs given by the function

$$P(\pi) = \sum_{i\in[1,m]:A_i\neq\emptyset} \left(\mu\Big(\sum_{d_j\in A_i} \ell(d_j)\Big)^{\alpha} + b\right). \qquad (1)$$

where μ is a tuning parameter, α reflects the superlinearity of the model, and b corresponds to the power consumed by an idle PM, i.e., when no load has been yet assigned to a PM. Finally, let us define the power requirements of a PM as the function $f(\cdot)$, such that $f(x) = 0$ if $x = 0$ and $f(x) = \mu x^{\alpha} + b$ otherwise. Then, the objective function is to minimize $P(\pi) = \sum_{i=1}^{m} f(\ell(A_i))$.

Based on this definition, the authors describe 4 different cases, $(\cdot,\cdot)$-VMA, $(C,\cdot)$-VMA, $(\cdot,m)$-VMA, and (C,m)-VMA. The difference between these 4 cases lies on whether the capacity and the number of PMs are bounded or not. The results presented in this work are related to the $(\cdot,m)$-VMA case, where the number of PMs is bounded but not their capacity; and the (C,m)-VMA case, where both the capacity and the number of PMs are bounded.

We also inherit the definition of optimal load, denoted as x^*, as the load at which the power rate, defined as the power consumed per unit of load and denoted by $\varphi(x) = f(x)/x$, is minimized. The optimal load of a PM is given by $x^* = (b/(\mu(\alpha-1)))^{1/\alpha}$. We refer the reader to [15] for further details on the description of the problem and the optimal load.

IV. Proposed Algorithms

As mentioned in Section II, we want to compare two online power aware algorithms, β-*VMA* and φ-*VMA*, with several online state-of-the-art allocation algorithms. In this section we introduce both algorithms and provide their pseudocode, which can be found in Algorithms 1 and 2.

The main constraint of online allocation algorithms is that decisions are to be taken as tasks arrive at the system, knowing nothing about the future tasks beforehand. Therefore, no optimality can be guaranteed, being only possible to provide upper and lower bounds on its competitive ratio, i.e. how much worse than an optimal clairvoyant are its results.

We start by describing β-*VMA*. The aim of power aware allocation algorithms is trying that the load in the PMs is as close as possible to x^*, in order to minimize the PM power consumption. β-*VMA* keeps the PMs load within $[x^*/\beta, 2x^*/\beta]$[3]. Upon its arrival, β-*VMA* assigns a VM d_i to a new PM when its load is larger than x^*/β; or to an existing PM s_j whose load $\ell(A_j) \leq \min\{x^*/\beta, C\}$, if any. In addition, the incoming task has to fit in the selected PM, i.e., $\ell(A_j) + \ell(d_i) \leq C$. If no PM meets these conditions, VM d_i is assigned to a new PM. Note that, in the $(\cdot,\cdot)$-VMA case, where $C = \infty$, there can only be one or zero PMs with $\ell(A_j) \leq \min\{x^*/\beta, C\}$.

, Parameter β allows the algorithm to be tuned. Arjona *et al.* [15] studied the competitive ratio of a particular version of this algorithm where $\beta = 2$, showing that better results can be obtained with this value than for lower β. However,

[3] β-*VMA* only fails to keep the load of a PM within $[x^*/\beta, 2x^*/\beta]$ when, for an incoming VM d_i, $\ell(d_i) > 2x^*/\beta$.

Algorithm 1: Online algorithm β-*VMA* for the $(\cdot,\cdot)$-VMA and $(C,\cdot)$-VMA problems.

for *each VM* d_i **do**
  if $\ell(d_i) > \min\{x^*/\beta, C\}$ **then**
    d_i is assigned to a new PM
  else
    d_i is assigned to any loaded PM s_j where $\ell(A_j) \leq \min\{x^*/\beta, C\}$ and $\ell(A_j) + \ell(d_i) \leq C$. If such loaded PM does not exist, d_i is assigned to a new PM.

Algorithm 2: Online algorithm φ-*VMA* for the $(\cdot,\cdot)$-VMA and $(C,\cdot)$-VMA problems.

for *each VM* d_i **do**
  if $\ell(d_i) \geq \min\{x^*, C\}$ **then**
    d_i is assigned to a new PM
  else
    d_i is assigned to machine s_k s.t. $k = \arg\min_{j:\ell(A_j)<\min\{x^*,C\}}\{\varphi(\ell(A_j) + \ell(d_i))\}$ and $\varphi(\ell(A_j) + \ell(d_i)) < \varphi(\ell(A_j))$. If such loaded PM does not exist, d_i is assigned to a new PM.

theoretical results do not necessarily imply better results in real deployments. For instance, in scenarios where the average load of the incoming VMs $\ell(d_i)$ is much smaller than x^*, values of β closer to 1 are expected to obtain better results, as the chances of the keeping the PM load close to X^* are higher. As the average expected d_i increases, more conservative configurations, i.e., larger values of β, will obtain better results.

Our second algorithm, *φ-VMA* takes the rate, φ, of a PM as reference. The rate φ is a convex function whose minimum value is obtained with a load $x = x^*$. *φ-VMA* assigns an incoming load d_i to a PM if these three conditions are met: (1) PM s_k load is below $\min\{x^*, C\}$ and the new VM d_i fits in the PM, i.e., $\ell(A_k) < \min\{x^*, C\}$ and $\ell(A_k) + \ell(d_i) < \min\{x^*, C\}$; (2) when adding the incoming load to the PM its rate decreases, i.e., $\ell(A_k) + \ell(d_i) < \varphi(\ell(A_j)$; (3) the selected PM is the one whose rate is minimal out of all the PMs meeting (1) and (2), i.e., $\varphi(\ell(A_j) + \ell(d_i))$ is minimal. If no such VM exists or $\ell(d_i) \geq \min\{x^*, C\}$, d_i is assigned to a new PM. *φ-VMA* allows that a PM is assigned more than x^* load if that reduces its rate, allowing more packed partitions than *β-VMA*.

Both *β-VMA* and *φ-VMA* have been extended to handle the cases where the number of PMs is bounded, namely $(\cdot, m)$-VMA and (C, m)-VMA. This is achieved by assigning the incoming task to the least loaded machine when no more new PMs are available.

V. Empirical Evaluation

In this section we evaluate the performance of *β-VMA* and *φ-VMA* via simulations, and compare them to other state-of-the-art online allocation algorithms.

A. Simulations Setup

The performance of both *β-VMA* and *φ-VMA* is first compared with a lower bound on the optimal power consumption, denoted *LBVMA*, that is obtained as follows. The input VMs are sorted in non-increasing order of their loads. Then, using this order, as many VMs as possible with load at least x^* are assigned to different PMs. Let L be total load of the VMs still unassigned. If there are $\lfloor L/x^* \rfloor$ still unused PMs they will be used. Otherwise all PMs will be used. Finally, the load L is assigned among all used PMs as if it could be infinitely divided (i.e., as a fluid), using a water-filling algorithm [20].

We evaluate both $(\cdot, m)$-VMA and (C, m)-VMA problems. Therefore, in the (C, m)-VMA case a VM can only be assigned to a PM if the latter has sufficient capacity to host it. We test both algorithms *β-VMA* and *φ-VMA* and also compare them with the following algorithms proposed in the literature:

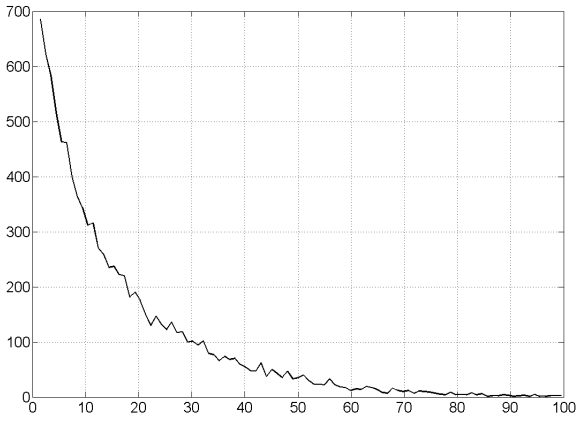
- Random Fit (RF) [27]: It chooses a PM for each VM uniformly at random among the PM. If the chosen PM cannot allocate the load of the VM, the process is repeated, until the VM is assigned to a PM.
- Next Fit (NF) [27]: Starting initially at the first PM, each new VM is assigned to the next PM after the latest PM to which a VM was assigned (in a cyclic fashion) and with sufficient capacity to host it.
- Least Full First (LFF) [27]: Each new VM is assigned to (one of) the least loaded PM(s) in the system with enough capacity to host it.
- Striping (S) [25]: Each new VM is assigned to (one of) the PM(s) with the smallest number of VMs assigned and with enough capacity to host it.
- Watts per Core (WC) [25]: Assigns each new VM to the PM whose power would suffer the smallest increase and with enough capacity to host it.
- First Fit (FF) [27]: Each new VM is placed in the first PM that can host it, starting from the first PM.
- Round Robin (RR) [25]: Like FF but, after the first VM is assigned, the search starts from the latest PM in which a VM was allocated.
- Packing (P) [25]: Each new VM is assigned to (one of) the PM(s) with the largest number of VMs assigned provided that the PM can host it.
- Most Full First (MFF) [27]: Each new VM is assigned to the most loaded PM in which it fits.

Observe that, given its nature, First Fit, Round Robin, Packing and Most Loaded First can be only considered for the (C, m)-VMA problem. If PMs had infinite capacity, these algorithms would place all VMs in only one PM. The remaining algorithms are evaluated for both $(\cdot, m)$-VMA and (C, m)-VMA.

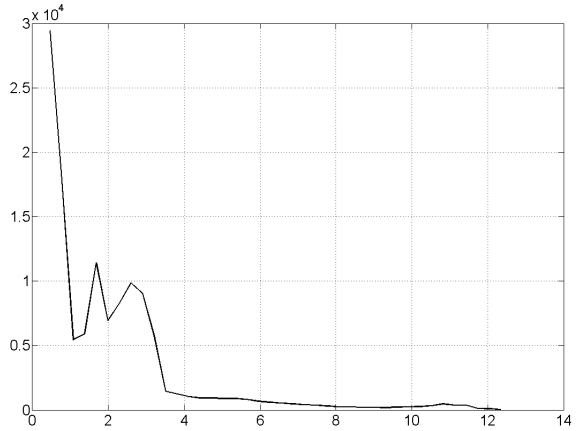
The behavior of the aforementioned algorithms is evaluated by inputting the two sets of traces, synthetic and real, shown in Figure 1. We call them *Trace A* and *Trace B*, respectively. Trace A is generated by randomly choosing the load of each VM following a power-law distribution with exponential cutoff, which has been chosen so 100% is the maximum task load of a VM. We randomly select 10000 integer loads using this distribution. The loads are inputed in the system sequentially. This leads us to the VM load distribution shown in Figure 1(a).

Trace B is obtained from public Google traces [24]. We extract all the tasks from these traces, assuming that each task is an independent VM. We assume that the VMs (tasks) arrive at the system in the same order at which hey arrived to the Google system, sorting them by the arrival time (given in the trace). The task load of a VM is the maximum CPU load of the task. The trace then contains 124885 VMs with loads varying between 0.31% and 12.5%. The resulting VM load distribution can be seen in Figure 1(b). The load values of the VMs for both distributions are given in percentage in order to scale them up depending on the maximum capacity of a PM in the (C, m)-VMA case or to an appropriate value in the $(\cdot, m)$-VMA case.

Each execution of the algorithms is run with a fixed number of PMs. This number of PMs increases from 1 to the number of VMs in the trace being used. This allows us to see how the power consumption and how the algorithms behavior evolve when the number of available PMs in the system varies. Finally, in order to evaluate both $(\cdot, m)$-VMA and (C, m)-VMA, we *emulate* different PMs by determining their α, b, μ and x^* parameters as well as the PM maximum capacity



(a) Trace A (synthetic traces)



(b) Trace B (Google traces)

Fig. 1. VMs load distributions used in the evaluations.

or the maximum task load when it corresponds. The PMs we emulate are 3 of the servers available in our laboratory, namely `Nemesis`, `Survivor` and `Erdos`, analyzed in [13], and for which we can easily compute the aforementioned parameters. In that study, all 3 servers were proven to follow superlinear power consumption models as the one described in Section III, having an optimal load x^* at which the efficiency of the server is maximized. We run the proposed algorithms for each one of these emulated PMs and compare the influence of the different values for these parameters on the final results.

Observe also that, in all our simulations, we assume that the servers being used are identical. We are aware that the cloud, or data centers in general, are not homogeneous, as shown in [24], [35], [30], [31]. However, we think that this approach does not imply a big loss of generality given that our algorithms depend on parameters derived from the servers themselves. Therefore, as long as these parameters are known (β, x^*, μ, α) these algorithms can easily be extended to heterogeneous clusters.

TABLE I
SIMULATION PARAMETERS FOR A SET OF MACHINES FOR THE $(\cdot, m)$-VMA CASE.

$(\cdot,\cdot)$-VMA case					
b	x^* [GCPS]	μ			
		$\alpha = 1.5$	$\alpha = 2$	$\alpha = 2.5$	$\alpha = 3$
73.05	10	1.46E-13	7.31E-19	4.87E-24	3.65E-29
92.15	30	3.55E-14	1.02E-19	3.94E-25	1.71E-30
111.25	50	1.99E-14	4.45E-20	1.33E-25	4.45E-31
135.125	75	1.32E-14	2.40E-20	5.85E-26	1.60E-31
159	100	1.01E-14	1.59E-20	3.35E-26	7.95E-32
187.65	130	8.01E-15	1.11E-20	2.05E-26	4.27E-32
206.75	150	7.12E-15	9.19E-21	1.58E-26	3.06E-32
350	300	4.26E-15	3.89E-21	4.73E-27	6.48E-33
541	500	3.06E-15	2.16E-21	2.04E-27	2.16E-33
779.75	750	2.40E-15	1.39E-21	1.07E-27	9.24E-34
1018.5	1000	2.04E-15	1.02E-21	6.79E-28	5.09E-34

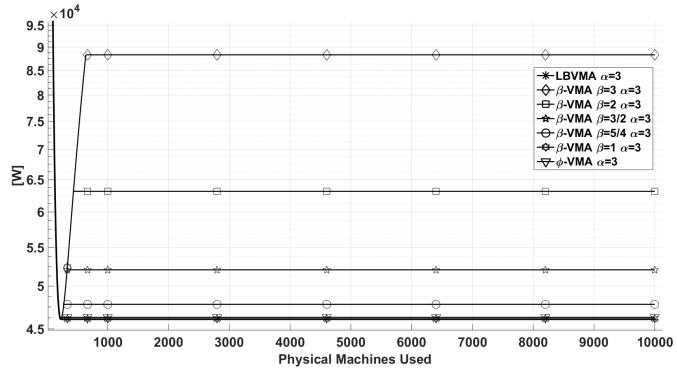
B. Results for $(\cdot, m)$-VMA

We first evaluate $(\cdot, m)$-VMA. The first step is to define the set of PMs that we are going to use to evaluate it. To do so, we fix the values of α, b and x^* and compute μ depending on the previous parameters. In particular, we used $\alpha = \{1.5, 2, 2.5, 3\}$ and $x^* = \{10, 30, 50, 75, 100, 130, 150, 300, 500, 750, 1000\}$ (given in (Giga)*Cycles per Second* (GCPS) following the conclusions from [13]). The values of b are determined by interpolation of the baseline costs of `Nemesis`, `Survivor` and `Erdos`, whose values for b ($\sim$ 85 W, 67 W and 215 W) are known from the experiments performed in [13]. These combinations of parameters result in 44 different instances of PMs which are shown in Table I.

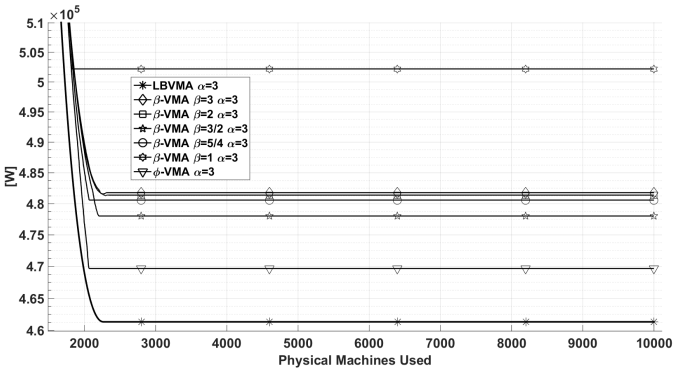
Additionally, taking advantage of the fact that the task loads from Trace A and B are given in percentage, and in order to study the importance of the x^* to task load ratio, we define the maximum VM load λ as the maximum task load that a VM arriving to the system can have. Therefore, the task load of the VMs arriving to the system will be the product of the task load (in percentage) and λ.

We study four different scenarios. In the first one we study the effect that using different values of β has on the performance of β-*VMA*. We also study the performance of β-*VMA* for different values of λ and for both Trace and Trace B. In addition, we also study the results that can be obtained by assigning the load based on the PM's rate, *i.e.* we use φ-*VMA* as a reference. Moreover, in this scenario, and in all the remaining ones, we also include the results of the lower bound *LBVMA*, to have a reference of the quality of the results achieved by the algorithms. The second scenario studies the effect of α for different values of λ and x^* when using β-*VMA*, φ-*VMA*, and *LBVMA*. Third and fourth scenarios are devoted to compare our proposed algorithms with the state-of-the-art algorithms, always keeping *LBVMA* as a reference. In the third scenario we study the relevance of λ while keeping α and x^* constant. Finally, in the fourth one, we study the effect of having different values of x^* while λ and α remain unaltered.

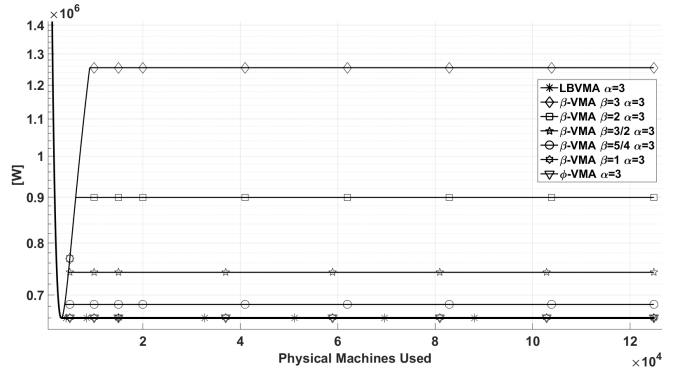
In scenario 1 we compare the power partitions obtained by β-*VMA* for different values of β comprehended between 1 and 3. To have an idea of the goodness of these results, they are



(a) Trace A, $\lambda = 10$ GCPS, $x^* = 75$ GCPS



(b) Trace A, $\lambda = 100$ GCPS, $x^* = 75$ GCPS



(c) Trace B, $\lambda = 100$ GCPS, $x^* = 75$ GCPS

Fig. 2. $(\cdot, m)$-VMA - Scenario 1: Comparing the power consumed by β-*VMA*, for different values of β for combinations of $x^* = 75$ GCPS, $\alpha = 2$ and $\lambda = \{10, 100\}$ GCPS for Trace A (Synthetic traces) and Trace B (Google traces). φ-*VMA* and $LBVMA$ are also shown for comparison.

plot jointly with the power consumed by the power partitions obtained with φ-*VMA* and *LBVMA*, that lower bounds the optimal power consumption. We also use different values of λ to observe the effects of the relation between the maximum load size and the optimal PM load. The results obtained are presented as graphs in which the power consumed is represented as a function of the number of PMs used. Similarly, for the sake of clarity, we do not show the power consumption resulting of using only one (or a few) machines and center the figure into more relevant cases.

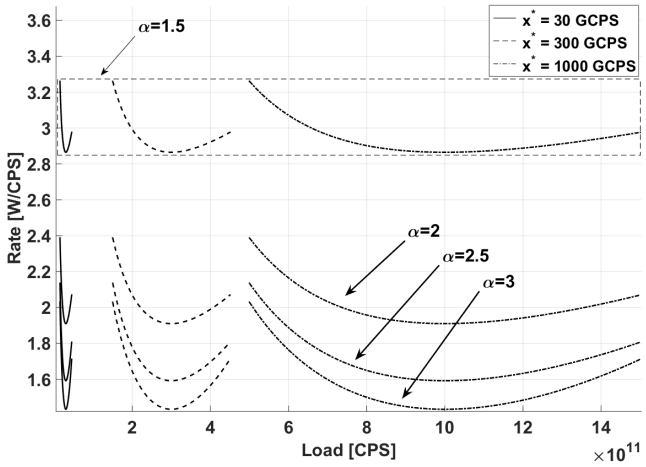


Fig. 3. Rates for different values of α and x^*.

Figure 2 shows the results for β-*VMA*, φ-*VMA*, and *LBVMA* in scenario 1 when a PM with $\alpha = 2$ and $x^* = 75$ GCPS is considered. The values of β used are $\beta = \{3, 2, 3/2, 5/4, 1\}$. The first thing we notice is that there is a dependence on the goodness of the results of β-*VMA* and the ratio λ/x^*, that gives us an approximate idea of the relative sizes of the VMs with respect to x^*. It can be noticed that, for Trace A, if $\lambda/x^* \ll 1$, *i.e.* VM sizes are small in comparison with x^*, the lower the β the better the β-*VMA* performance, as can be seen in Figure 2(a). This comes as a consequence of a better ability of the algorithm for loading the PMs up to a value which is close to x^*. On the other hand, as shown in Figure 2(b), when $\lambda/x^* \geq 1$, low values of β lead to configurations where the load in a PM can be much larger than x^* and, hence, result in a power consumption substantially larger. This intuition is reinforced by the results observed in Figure 2(c), where $\beta = 1$ achieves the best results despite of having $\lambda/x^* > 1$. Trace B VMs are much smaller, in percentage, than Trace A VMs. Therefore, even when $\lambda/x^* > 1$, most of the loads will be hardly comparable in size with x^*, what leads to more granularity and better results for lower values of β. As a consequence, β-*VMA* obtains its best results when $\beta = 1$ as the algorithm is able to keep the load per PM in the neighborhood of x^*. The main conclusion to be extracted from β-*VMA* in scenario 1 is the strong dependence of β-*VMA* on the expected size of loads to be allocated.

On the other hand, we have φ-*VMA* which obtains partitions which are consistently close to the ones obtained by *LBVMA* and, in most cases, the least power consuming ones. This can be explained by how the algorithm assigns workloads to the PMs. In Figure V-B we see different curves, each one representing the rate of different PMs from $0.5x^*$ load to $1.5x^*$. We can group them in 3 sets of 4 curves from left to right, each one of these curves corresponding to $\alpha = \{1.5, 2, 2.5, 3\}$ and each

set to $x^* = \{30, 300, 1000\}$ GCPS[4]. It can be seen that the rate of each group exhibits a different behavior, having a steeper slope after the x^* load for smaller values of x^*. Depending on this slope and on the the size of the VMs, φ-*VMA* allows the load assigned to the PM to exceed x^* by a larger margin, mimicking β-*VMA* with $\beta \leq 1$; or not exceeding it at all, mimicking the behavior of β-*VMA* with larger values of β.

To reduce the number of curves in the following scenarios, we will only show β-*VMA* for $\beta = \{3/2, 1\}$. These values are the ones for which β-*VMA* obtained the best results in the experiments shown in Figure 2, and will give a good intuition of the goodness of the results that can be achieved by using β-*VMA*.

Scenario 2 compares the power consumed by partitions obtained with β-*VMA* or φ-*VMA* and for Trace A and Trace B for different values of α and x^*, as well as for Trace A and B. We also compare these results to the ones achieved by *LBVMA*.

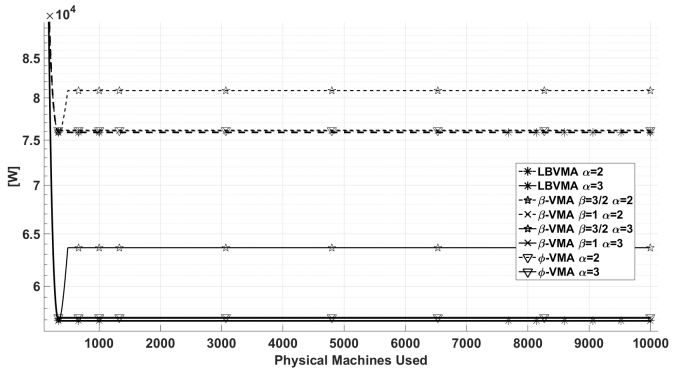
Figure 4 shows the results for Trace A, $x^* = 50$ GCPS, and for 2 values of λ, 10 and 100 GCPS. We can clearly see how the power consumption is smaller for larger values of α once the optimal number of used PMs is reached. This is mainly conditioned by how μ decreases as α increases (See Table I). Also, as it can be observed, there is no qualitative difference in the solutions when α varies for a given configuration. Due to space limitations we only show a small sample of the results for Trace A and B, however, similar results are obtained for other combinations of x^* and λ.

Regarding the performance of the algorithms, we can see how the power consumed by the partitions found with φ-*VMA* is consistently close to the ones obtained by *LBVMA*, in all cases. However, despite this closeness, β-*VMA* with $\beta = 1$ is still able to slightly improve its results depending on the relative size of the loads. This can be a consequence of the ratio λ/x^*, like in Figure 4(a); or when most of the loads are much smaller than x^*, as shown in Figure 4(c) for Trace B. In Figure 4(b), though, where VMs can have a size comparable to x^*, φ-*VMA* clearly outperforms the results achieved by the instances of β-*VMA*.

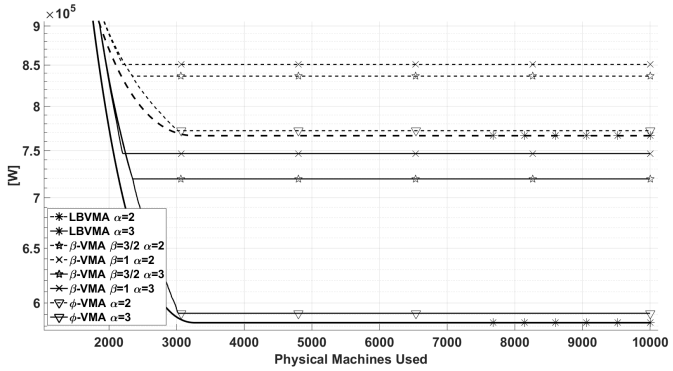
Scenario 3 compares the performance of *LBVMA*, φ-*VMA* and β-*VMA* with the other assignment algorithms proposed in the literature. Here, the values of x^* and α are fixed to 50 GCPS and 2, respectively, while the value of λ varies. In particular we use $\lambda = \{10, 30, 100\}$ GCPS. Figures 5 and 6 present the results for Trace A and Trace B.

We can easily see, in general, 3 different trends in Figures 5 and 6. The first trend would include *LBVMA*, φ-*VMA* and β-*VMA*; then we have a second one including *Striping*, *RF*, *NF* and *LLF*; and, finally, in some sort of no-man's land, we have *WC*. These trends have their origin in power awareness. While *LBVMA*, φ-*VMA*, β-*VMA* and *WC* are power aware, the rest are not. According to Figures 5 and 6, power aware algorithms clearly outperform the non power aware ones.

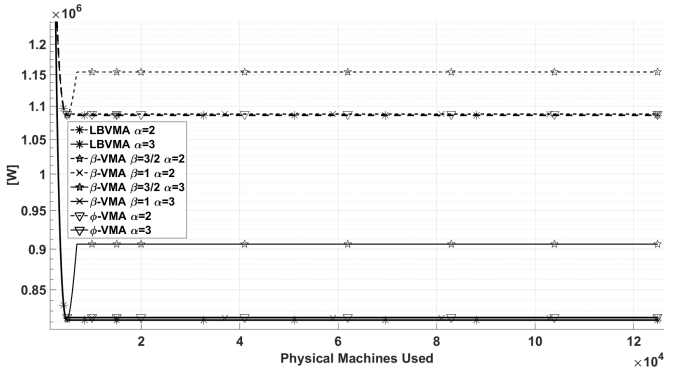
[4]Remember that a PM is defined by a value of α, b and μ. However, because of how we are emulating the PMs in our experiments, b and μ are obtained from x^*.



(a) Trace A, $\lambda = 10$ GCPS, $x^* = 50$ GCPS



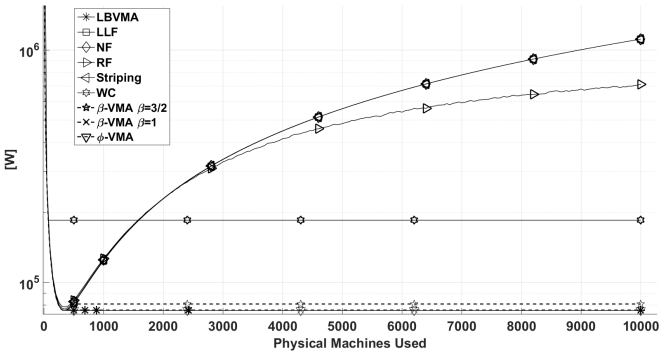
(b) Trace A, $\lambda = 100$ GCPS, $x^* = 50$ GCPS



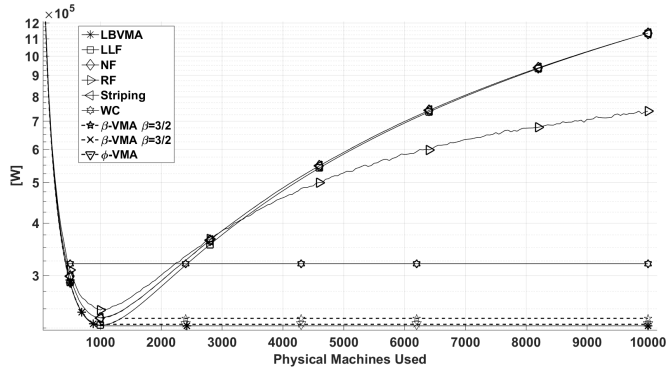
(c) Trace B, $\lambda = 100$ GCPS, $x^* = 50$ GCPS

Fig. 4. $(\cdot, m)$-VMA - Scenario 2: Comparing the power consumed by β-*VMA*, for different values of β, and φ-*VMA* with the lower bound $LBVMA$ for combinations of $x^* = 50$ GCPS, $\alpha = \{2, 3\}$ and $\lambda = \{10, 100\}$ GCPS for Trace A (Synthetic traces) and Trace B (Google traces).

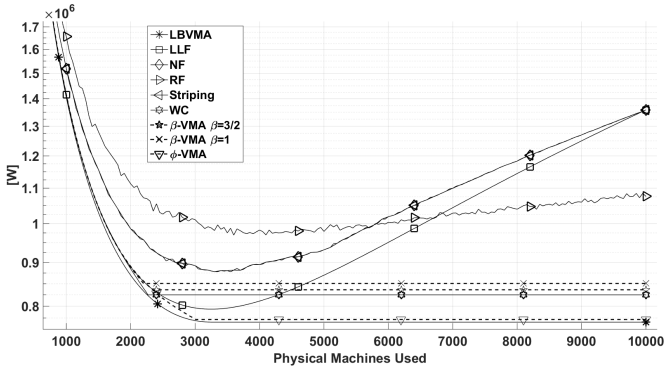
It is interesting to see how *WC* reduces its gap with *LBVMA* (for Trace A) as the ratio λ/x^* increases and even performs better than β-*VMA* when $\lambda = 100$ GCPS although remains far from φ-*VMA*. This does not happen, though, for Trace B due to the smaller average VM load, that gives advantage to β-*VMA* and φ-*VMA*. With Trace B, due to its nature, *WC* obtains partitions that use less PMs and hence, because of the superlinear dependence of the power consumption on the load, have higher power consumption. This can also be observed in the figures, where *WC* reaches the steady state earlier than the



(a) $\lambda = 10$ GCPS

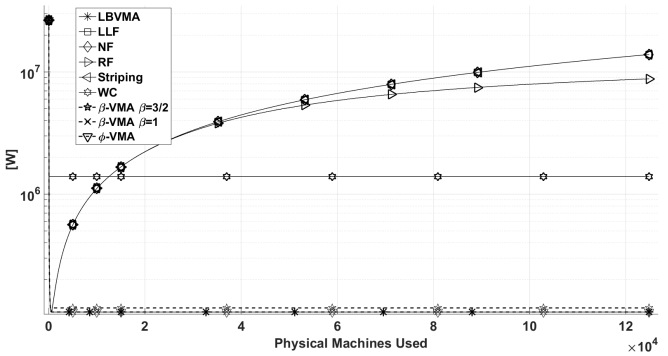


(b) $\lambda = 30$ GCPS

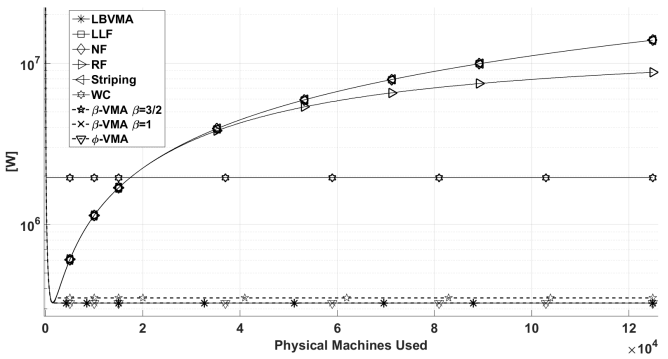


(c) $\lambda = 100$ GCPS

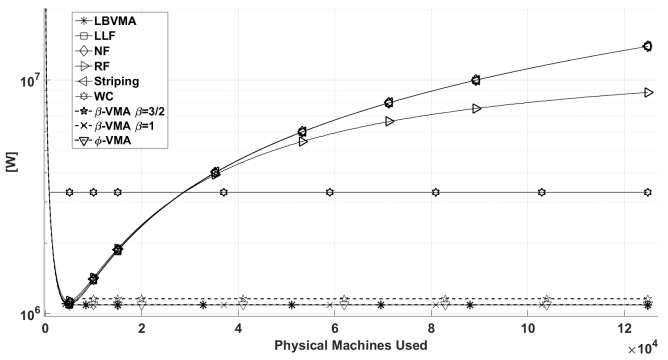
Fig. 5. $(\cdot, m)$-VMA - Scenario 3: Comparing the power consumed by the different assignment algorithms for $x^* = 50$ GCPS, $\alpha = 2$ and $\lambda = \{10, 30, 100\}$ GCPS for Trace A (Synthetic traces).



(a) $\lambda = 10$ GCPS



(b) $\lambda = 30$ GCPS



(c) $\lambda = 100$ GCPS

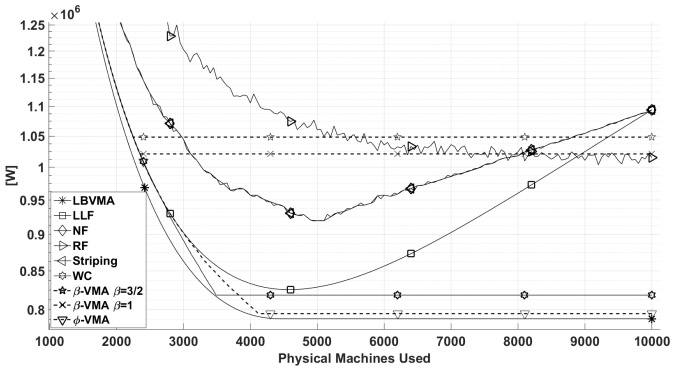
Fig. 6. $(\cdot, m)$-VMA - Scenario 3: Comparing the power consumed by the different assignment algorithms for $x^* = 50$ GCPS, $\alpha = 2$ and $\lambda = \{10, 30, 100\}$ GCPS for Trace B (Google traces).

other power aware algorithms.

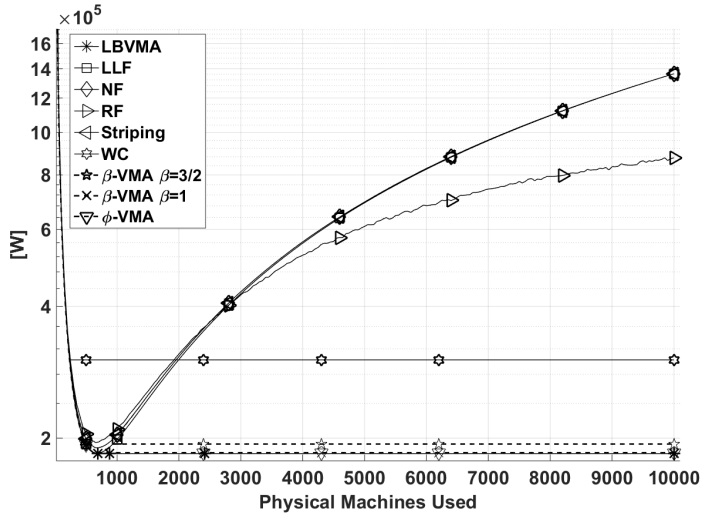
Similarly, we can observe that the results in Figure 5(c) are tighter. This is a consequence, again, of the high value of λ/x^*, resulting in many PMs allocating a small number of VMs. In fact, the steady state for φ-*VMA*, β-*VMA*, *WC* and even for *LBVMA* is reached between the 2000 and the 3000 PMs while in the previous cases was reached before the 1500 PMs. Again, this behavior is not replicated with Trace B due to its lower average task load.

Note also that the non power aware algorithms pay a higher power bill due to the use of a larger amount of PMs (in general) with a smaller amount of load per PM, resulting in a very inefficient usage of the available resources. This behavior is consistent for both Traces A and B.

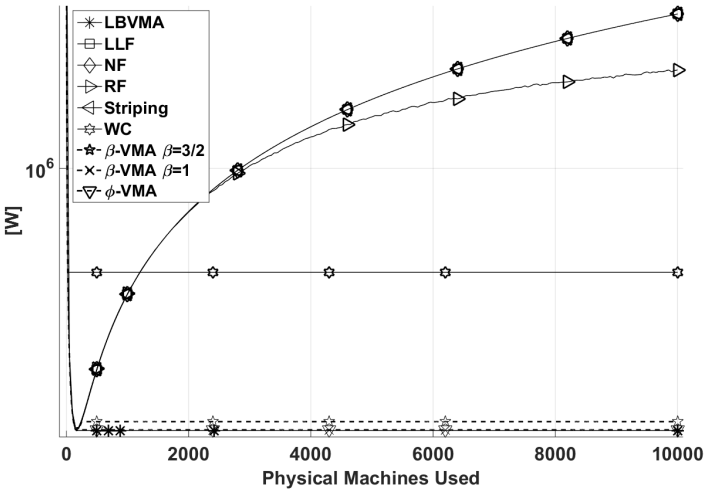
Finally, observe how the smaller the ratio λ/x^*, the larger the gap between our proposed algorithms, φ-*VMA* and β-*VMA*, and the other ones. φ-*VMA* exhibits again the best results in average, being always close to *LBVMA*, independently of the ratio λ/x^* and thanks to the advantage of basing its assignment on the rate ρ. It is also important to note than, even when φ-*VMA* obtains partitions that use less PMs than β-*VMA*, it is less aggressive in reducing the number of PMs



(a) $x^* = 10$

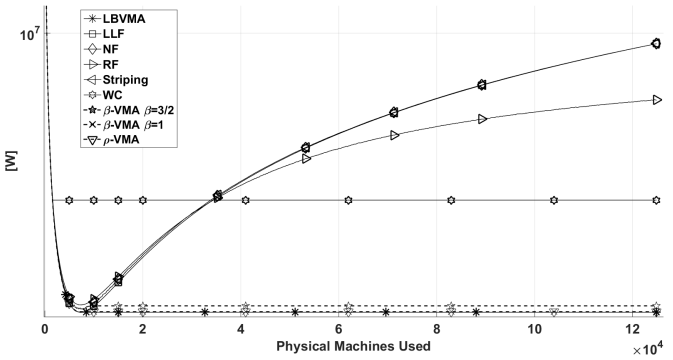


(b) $x^* = 75$

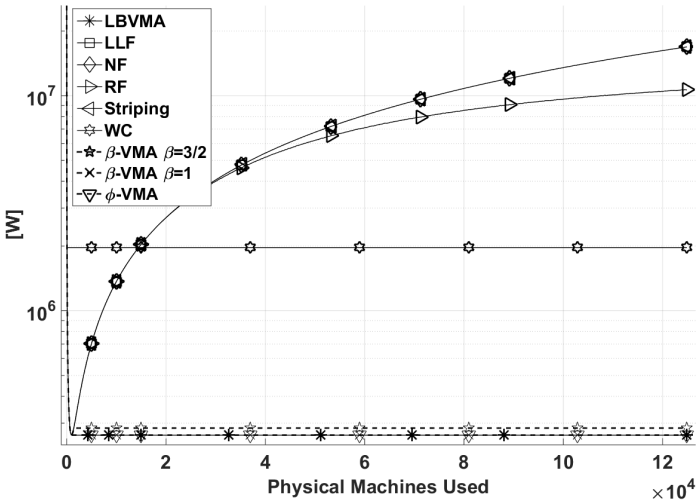


(c) $x^* = 300$

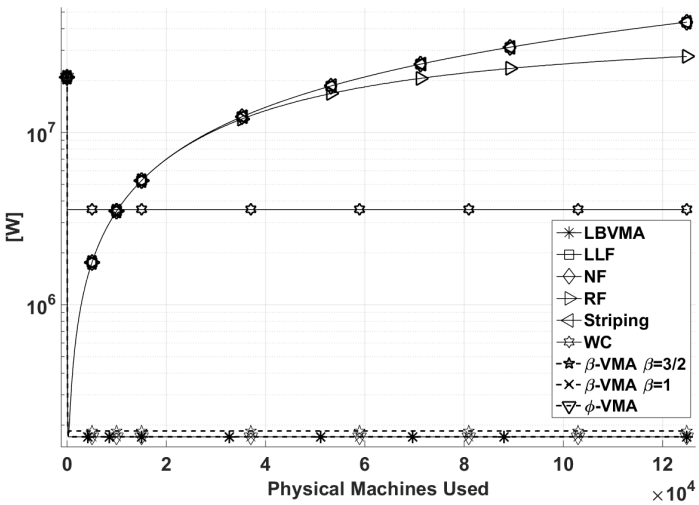
Fig. 7. $(\cdot, m)$-VMA - Scenario 4: Comparing the power consumed by the different assignment algorithms for $\lambda = 30$ GCPS, $\alpha = 2$ and increasing values of x^* for Trace A (Synthetic traces).



(a) $x^* = 10$



(b) $x^* = 75$



(c) $x^* = 300$

Fig. 8. $(\cdot, m)$-VMA - Scenario 4: Comparing the power consumed by the different assignment algorithms for $\lambda = 30$ GCPS, $\alpha = 2$ and increasing values of x^* for Trace B (Google traces).

than *WC*, thanks to its rate-awareness, obtaining, hence, more efficient results.

Let us now analyze the results of the last scenario. Here we keep $\lambda = 30$ GCPS and $\alpha = 2$ constant while we vary the value of x^*. These results are shown in Figure 7 for Trace A and in Figure 8 for Trace B.

The results are similar for both traces. It can be seen that, for Trace A and the smallest value of x^*, $x^* = 10$ GCPS, we obtain the tighter results for all algorithms. Observe how, due to the small value of the ratio λ/x^*, β-*VMA* obtains some of the worse results in this case. In the other cases, as the ratio λ/x^* decreases, the results obtained by β-*VMA* get closer to φ-*VMA* and better than the ones achieved by *WC*, *LLF*, *NF*, *Striping*, and *RF*, whose results get away from *LBVMA* as x^* increases, leading to partitions with a larger power consumption, oppositely to our algorithms, that stay close to the lower bound. Although we do not show them, for the sake of saving space, the figures for $x^* = \{30, 50, 100, \ldots\}$, for both traces A and B, would complete this gradual process, being the intermediate steps between the already shown ones. These results are in line with the results from Scenario 3 and are motivated by the fact that the state-of-the-art algorithms tend to use a larger amount of PMs keeping its average load low and, hence, paying a high price because of the b parameter. This, however, is not the case of *WC*, which, on the other hand, obtains a more packed partition, loading PMs beyond x^* and paying an extra cost due to the superlinearity of the power consumption with respect to the load. Finally, observe that, as happened in the previous scenarios, φ-*VMA* and β-*VMA* (when $\beta = 1$) achieve the best results, with the exception of the case of $x^* = 10$ GCPS for Trace A due to the value of the ratio λ/x^*.

C. Results for (C, m)-VMA

As we did with $(\cdot, m)$-VMA in Subsection V-B, we start by defining the set of PMs we are going to work with. While for $(\cdot, m)$-VMA we assumed that PMs had infinite capacity, in (C, m)-VMA the PMs capacity is bounded. We denote the capacity as C. We define 2 sets of instances that we name after 2 real PMs from our laboratory, `Nemesis` and `Erdos`. We use, as a reference, their maximum capacity, 11.2 and 153.6 GCPS; and idle cost b, 80 and 200 W. Jointly with C and b, we use $\alpha = \{1.5, 2, 2.5, 3\}$, and $x^* = \{0.5, 0.65, 0.75, 0.9, 1, 1.1\} \cdot C$. We can now compute the value of μ for each combination of these 4 parameters fully defining, then, our set of PMs, which is shown in Tables III and II.

In this case we consider 3 different scenarios. In the first scenario we study, like for $(\cdot, m)$-VMA, the effect that different values of β have on β-*VMA* as well as the effect that different PMs has on φ-*VMA*. In scenario 2, we compare again β-*VMA* and φ-*VMA* with *LBVMA* for different values of α to evaluate its influence on the power partitions when capacity is considered. In the third scenario, we evaluate the performance of β-*VMA*, φ-*VMA*, and all the state-of-the-art algorithms when α, b and C are fixed and x^* varies. Note that all simulations are run for both Trace A and Trace B as well as for both families of PMs, however, due to space limitations only the most relevant results are shown. Similarly, observe

TABLE II
SIMULATION PARAMETERS FOR A SET OF MACHINES WITH b AND C SIMILAR TO ERDOS.

`Erdos`-like Family		Max. Capacity $C = 153.6$ GCPS, $b = 200$ W, μ			
x^* [GCPS]	$x^*[\%C]$	$\alpha = 1.5$	$\alpha = 2$	$\alpha = 2.5$	$\alpha = 3$
76.8	50	1.88E-14	3.39E-20	8.16E-26	2.21E-31
99.84	65	1.27E-14	2.01E-20	4.23E-26	1.00E-31
115.2	75	1.02E-14	1.51E-20	2.96E-26	6.54E-32
138.24	90	7.78E-15	1.05E-20	1.88E-26	3.79E-32
153.6	100	6.64E-15	8.48E-21	1.44E-26	2.76E-32
168.96	110	5.76E-15	7.01E-21	1.14E-26	2.07E-32

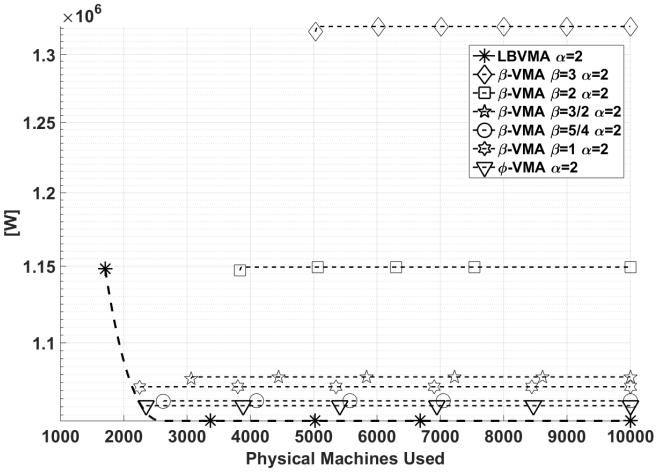
TABLE III
SIMULATION PARAMETERS FOR A SET OF MACHINES WITH b AND C SIMILAR TO NEMESIS.

`Nemesis`-like Family		Max. Capacity $C = 11.2$ GCPS, $b = 80$ W, μ			
x^* [GCPS]	$x^*[\%C]$	$\alpha = 1.5$	$\alpha = 2$	$\alpha = 2.5$	$\alpha = 3$
5.6	50	3.82E-13	2.55E-18	2.27E-23	2.28E-28
7.28	65	2.58E-13	1.51E-18	1.18E-23	1.04E-28
8.4	75	2.08E-13	1.13E-18	8.25E-24	6.75E-29
10.08	90	1.58E-13	7.87E-19	5.23E-24	3.91E-29
11.2	100	1.35E-13	6.38E-19	4.02E-24	2.85E-29
12.32	110	1.17E-13	5.27E-19	3.17E-24	2.14E-29

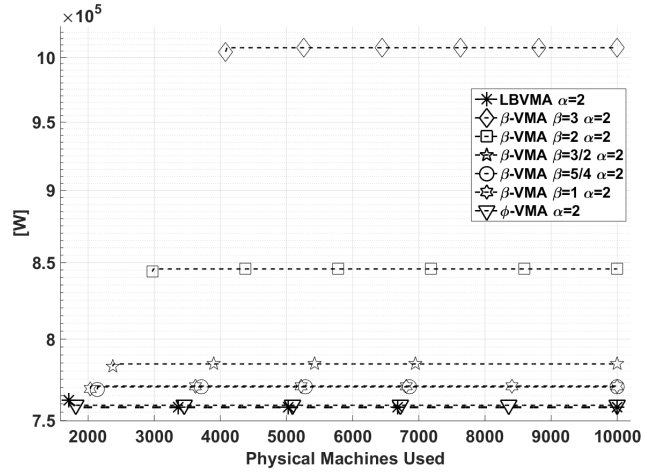
that results are presented, again, as graphs in which the power consumed is represented as a function of the number of PMs used but, this time, these results do not start from 1 PM. Each one of the results of the different algorithms starts from the number of PMs for which it obtained the first valid solution where PMs do not have to be loaded beyond their capacity C.

In Figure 9 we can see some of the results for scenario 1 when a Nemesis-like PM is used, what implies a capacity $C = 11.2$ GCPS. We show the results for $x^* = \{0.65, 0.9\} \cdot C$ and Trace A in Figures 9(a) and 9(b), and for Trace B and $x^* = 0.9 \cdot C$ in Figure 9(c). Interesting remarks about β-*VMA* are, for instance, how the results when $\beta = 1$ are worse than for $\beta = 5/4$ for low values of xx^*, as shown in Figure 9(a), and how they get better as x^* increases, as can be seen in Figure 9(b). This is, as it was with $(\cdot, m)$-VMA, a consequence of the ratio λ/x^*. However, the intuition now is opposed to the one in the $(\cdot, m)$-VMA case. For $(\cdot, m)$-VMA, β-*VMA* with $\beta = 1$ improved for smaller ratios λ/x^* because smaller VM sizes allowed β-*VMA* to do a better load balancing around x^*, while for larger ratios, larger VMs could lead to PMs loaded far beyond x^*. For (C, m)-VMA[5], due to the fact that we cannot exceed the capacity of the PM, going far beyond x^* is not feasible. Moreover, a value of x^* which is close to C would improve the packing abilities of β-*VMA* as the biggest loads of the load distribution would not penalize the power consumption as they do then x^* has smaller values, such as in the case of $x^* = 0.5 \cdot C$. This intuition is reinforced with the results obtained with Trace B where, due to the small size of the VMs, β-*VMA* with $\beta = 1$ almosts matches the results obtained with *LBVMA*.

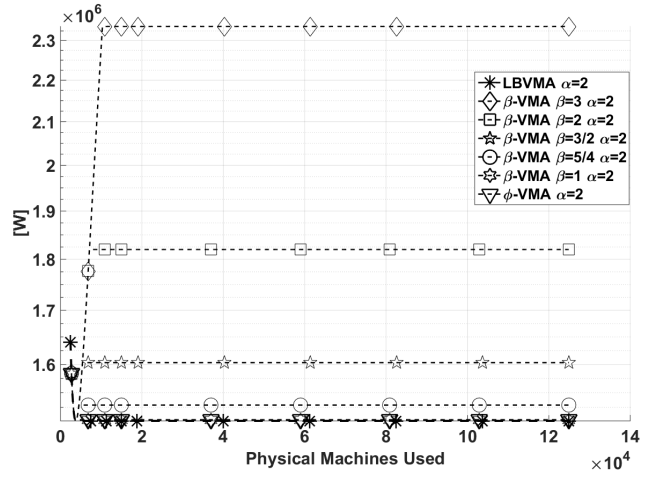
[5]Observe that for the (C, m)-VMA case $\lambda = C$.



(a) Trace A, $C = 11.2$ GCPS, $x^* = 0.65 \cdot C$



(b) Trace A, $C = 11.2$ GCPS, $x^* = 0.9 \cdot C$



(c) Trace B, $C = 11.2$ GCPS, $x^* = 0.65 \cdot C$

Fig. 9. (C, m)-VMA- Scenario 1: Comparing the power consumed by β-*VMA*, for different values of β, and φ-*VMA* with the lower bound $LBVMA$ in Nemesis, with $C = 11.2$ GCPS, for $x^* = \{0.65, 0.9\} \cdot C$, $\alpha = 2$ and for Trace A (Synthetic traces) and Trace B (Google traces).

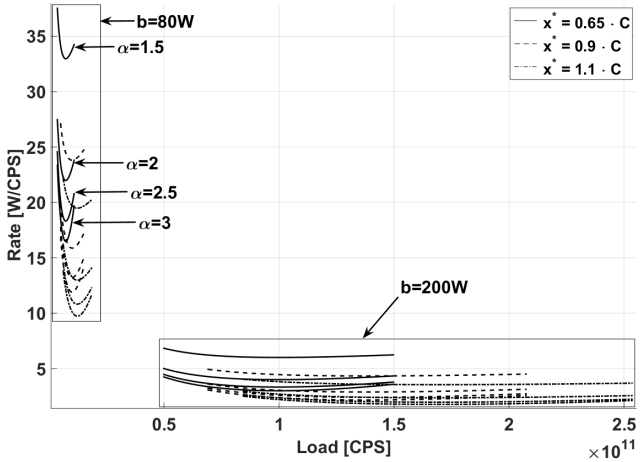


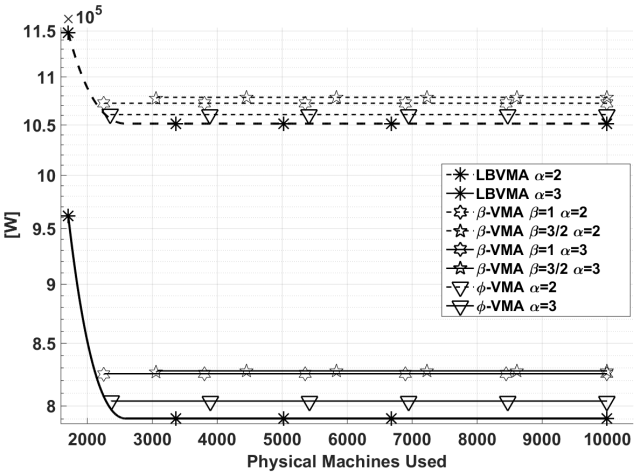
Fig. 10. Rates for different values of α and x^*.

Regarding φ-*VMA*, Figure V-C shows different rates for different PMs (the groups of lines that can be seen from left to right for $b = \{80, 200\}$), the different values of α, and different values of x^*. Each curve varies between $0.5x^*$ and $1.5x^*$. Again, we can see how for different PMs we have different slopes before and after x^*. This steepness depends mainly on the type of PM and implies that sometimes will be better not to assign load to a PM beyond x^*, while in some other it will be affordable. For this reason, as can be seen in the different subfigures of Figure V-B, being rate aware makes, again, a difference, and φ-*VMA* consistently outperforms β-*VMA*.

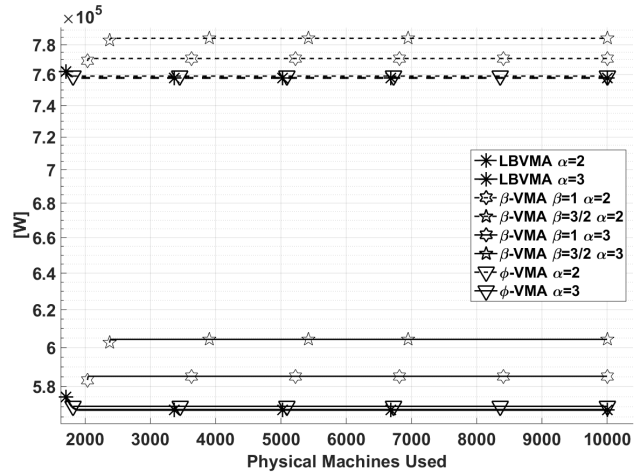
In the remaining scenarios, for the sake of improving the readability of the figures, we will only show the results of β-*VMA* with $\beta = \{3/2, 1\}$.

The results for the second scenario, shown in Figures 11, throw very similar results to the ones obtained for $(\cdot, m)$-VMA. φ-*VMA* performance is again very close to *LBVMA* when it does not match it. Similarly, β-*VMA* is again worse than φ-*VMA*, whose rate-awareness advantage makes a bigger difference than in the $(\cdot, m)$-VMA case. Observe that we show two different values of x^* and different Traces to show the consistency of the results. As for $(\cdot, m)$-VMA, there is no qualitative difference in the solutions when α varies for a given configuration.

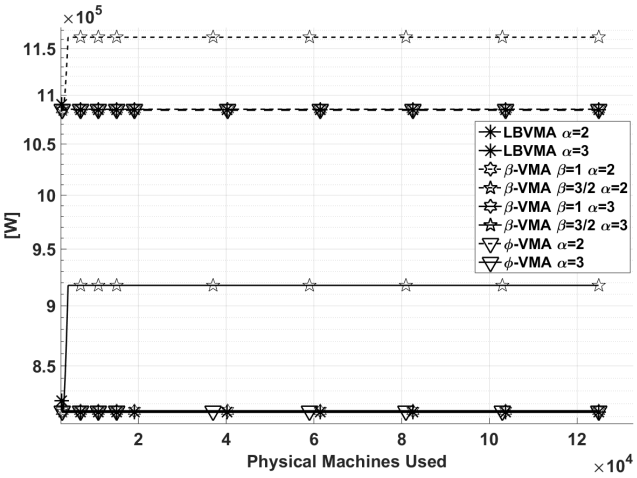
Figure 12 and Figure 13 show the results obtained for the third scenario for Traces A and B, respectively. In them we compare the performance of β-*VMA*, φ-*VMA*, *LBVMA*, *FF*, *MLF*, *Packing*, *RR*, *RF*, *LFF*, *NF*, *Striping* and *WC* for PMs such as the ones defined in Table II. We can easily observe, independently of the trace used, that *RP*, *LFF*, *NF* and *Striping* consistently obtain partitions which result in a higher cost in Watts than the other algorithms. The nature of this set of algorithms implies using a large number of PMs with a small amount of load per PM, resulting, always, in a higher power consumption. For this reason, and for the sake of clarity when



(a) Trace A, $C = 153.6$, $x^* = 0.65 \cdot C$

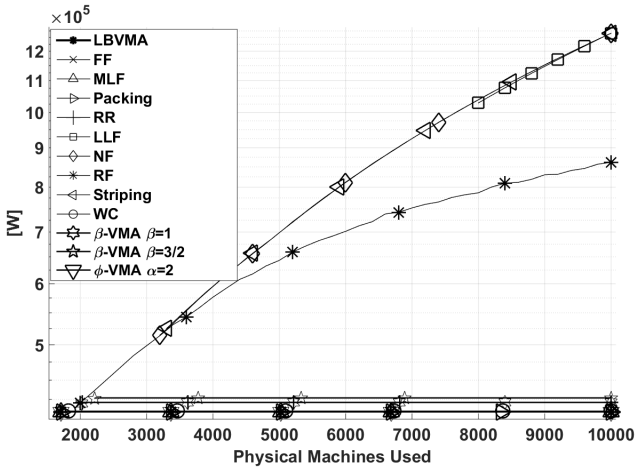


(b) Trace A, $C = 153.6$, $x^* = 0.9 \cdot C$

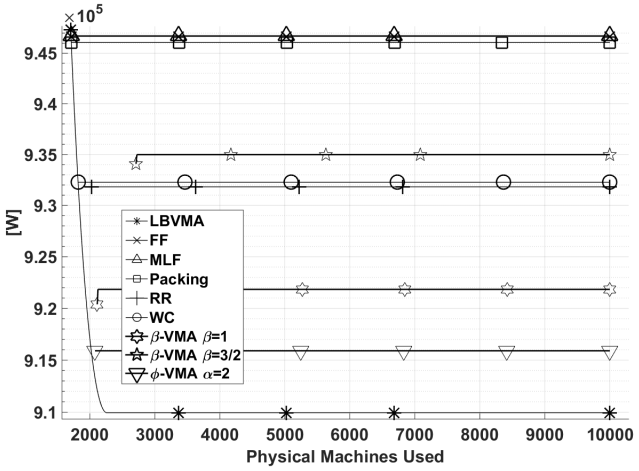


(c) Trace B, $C = 153.6$, $x^* = 0.65 \cdot C$

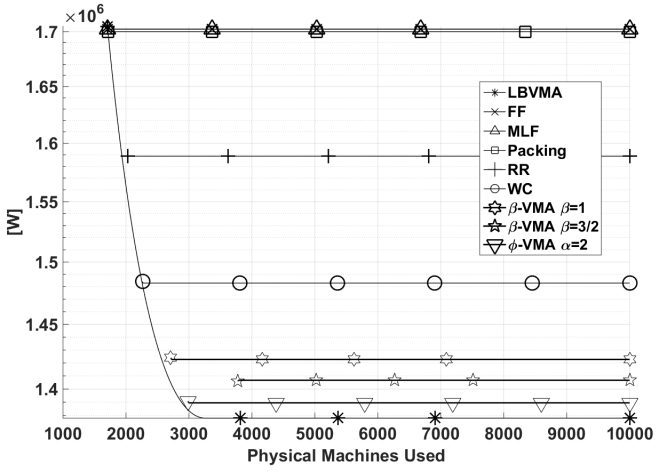
Fig. 11. (C, m)-VMA - Scenario 2: Comparing the power consumed by *φ-VMA*,*β-VMA*, with $\beta = \{1, 3/2\}$, and *LBVMA* for $x^* = \{0.65, 0.9\} \cdot C$, $C = 153.6$ GCPS, $\alpha = \{2, 3\}$ for Traces A and B.



(a) $x^* = 1$

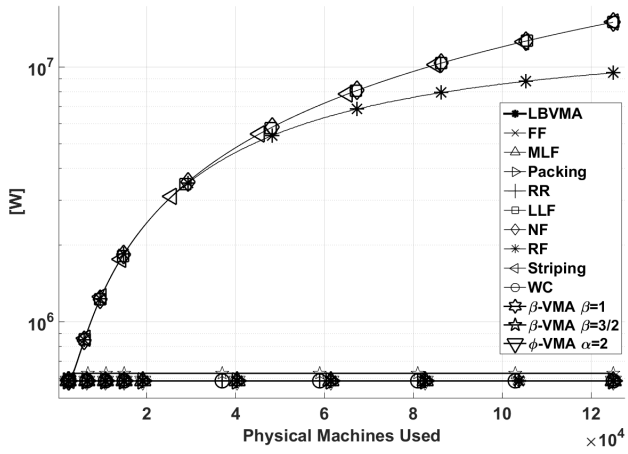


(b) $x^* = 0.75$

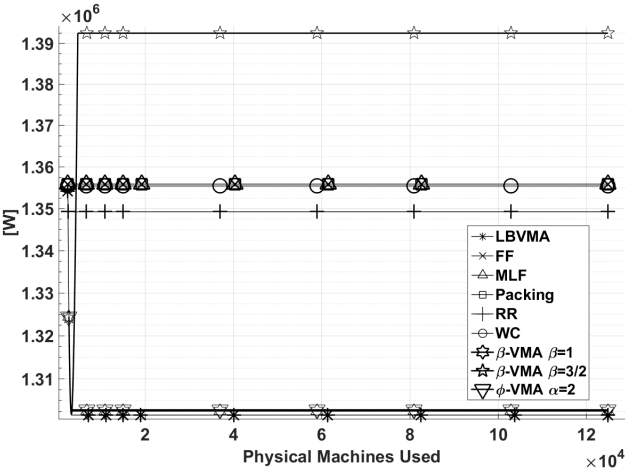


(c) $x^* = 0.5$

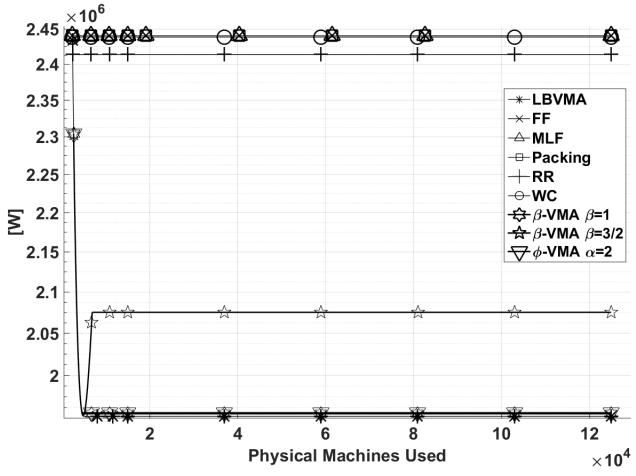
Fig. 12. (C, m)-VMA - Scenario 3: Comparing the power consumed by the different assignment algorithms for $C = 11.2$ GCPS, $\alpha = 2$, $\beta = \{1, 3/2\}$ and different values of x^* for Trace A (Synthetic traces).



(a) $x^* = 1$



(b) $x^* = 0.75$



(c) $x^* = 0.5$

Fig. 13. (C, m)-VMA - Scenario 3: Comparing the power consumed by the different assignment algorithms for $C = 11.2$ GCPS, $\alpha = 2$, $\beta = \{1, 3/2\}$ and different values of x^* for Trace B (Google traces).

plotting and comparing the other algorithms, we only show *RP*, *LFF*, *NF* and *Striping* in subfigures 12(a), and 13(a) as an example. The rest of the subfigures from Figure 12 and Figure 13 zoom in the results of the other algorithms.

Oppositely to the $(\cdot, m)$-VMA case, in (C, m)-VMA *WC* exhibits a better performance than both β-*VMA* when $x^* = \{1, 0.75\} \cdot C$. In fact, β-*VMA* with $\beta = 3/2$ also loses in the comparison with the group of bin-packing-like algorithms formed by *FF*, *MLF*, *Packing* and *RR* in every case except for $x^* = 0.5 \cdot C$, shown in Figures 12(c) and 13(c). This worse performance of β-*VMA* when $\beta = 3/2$ is a consequence of the ability of the other algorithms to obtain more packed solutions whose load is, additionally, closer to x^* than the $\beta = 3/2$ solution.

On the other hand, φ-*VMA* outperforms or matches the results of *FF*, *MLF*, *Packing* and *RR* for Trace A and B. φ-*VMA* obtains the best results in all cases except for the ones in Figures 13(b) and 13(c), where β-*VMA* with $\beta = 1$ achieves results which are slightly better than φ-*VMA*'s.

VI. Conclusions

In this paper we have proposed two novel allocation algorithms and compared them with multiple state-of-the-art allocation algorithms empirically. We have proved, by simulation, that the both algorithms we propose consistently obtain better results that other algorithms of different nature (packing algorithms, power aware, non power aware, ...) in two of the particular cases of the virtual machine allocation problem, being one of them, (C, m)-VMA, the closest one to real environments. One of them in particular, φ-*VMA*, has proved itself to be the most adaptable one, obtaining most of the best results for different configurations and for both $(\cdot, m)$-VMA and (C, m)-VMA.

Our future work is divided in two branches, the theoretical and the empirical one. On the theoretical [15] side, there are two aspects we plan to include and that substantially increase the complexity of the $(\cdot, \cdot)$-VMA problem. The first one is considering the possibility that the load incurred by a VM changes over time, what can improve or degrade the quality of the chosen placement. The second is that the assignment of VMs to PMs is not final (and VMs can migrate, maybe at a cost). In fact, if such migration cost is small or even free, our offline positive results can also be used in these new models, since an offline approximation algorithm can be run each time a load changes or a new VM arrives. Although nowadays there is such a cost, the irruption of technologies such as Docker containers have largely reduced this cost for stateless virtualized applications.

Regarding the empirical, having outperformed φ-*VMA*, and also β-*VMA*, algorithms which are currently being used in popular cloud platforms, such as Openstack or OpenNebula, we firmly think that, at least φ-*VMA*, is ready to be deployed in such or similar platforms (probably Openstack via Openstack Neat [19]), and tested in both homogeneous and heterogeneous clusters, which would be the natural extension to this work.

ACKNOWLEDGEMENTS

This work has been supported in part by MINECO grant TEC2014- 55713-R, Regional Government of Madrid (CM) grant Cloud4BigData (S2013/ICE-2894, co- funded by FSE & FEDER), NSF of China grant 61520106005, and EC H2020 grants ReCred and NOTRE.

REFERENCES

[1] Amazon quarterly results. http://phx.corporate-ir.net/phoenix.zhtml?c=97664&p=irol-reportsother. Accessed January 23, 2016.

[2] Amazon web services. http://aws.amazon.com. Accessed January 23, 2016.

[3] Citrix. http://www.citrix.com. Accessed January 23, 2016.

[4] Citrix quarterly results. https://www.citrix.com/news/financial-releases.html. Accessed January 23, 2016.

[5] Eucalyptus scheduling policies. http://docs.hpcloud.com/eucalyptus/4.2.1/#install-guide/sched_pol.html. Accessed February 10, 2016.

[6] Microsoft azure. https://azure.microsoft.com. Accessed April 29, 2016.

[7] Opennebula scheduling policies. http://docs.opennebula.org/4.14/administration/references/schg.html. Accessed February 10, 2016.

[8] Openstack. https://www.openstack.org/. Accessed January 23, 2016.

[9] Openstack scheduling policies. http://docs.openstack.org/kilo/config-reference/content/section_compute-scheduler.html. Accessed February 10, 2016.

[10] Rackspace. http://www.rackspace.com. Accessed January 23, 2016.

[11] Rackspace quarterly results. http://ir.rackspace.com/phoenix.zhtml?c=221673&p=quarterlyearnings. Accessed January 23, 2016.

[12] Amazon. Cloud computing, server utilization, & the environment. https://aws.amazon.com/es/blogs/aws/cloud-computing-server-utilization-the-environment/. Accessed January 23, 2016.

[13] Jordi Arjona Aroca, Angelos Chatzipapas, Antonio Fernandez Anta, and Vincenzo Mancuso. A measurement-based characterization of the energy consumption in data center servers. *Selected Areas in Communications, IEEE Journal on*, 33(12):2863–2877, 2015.

[14] Jordi Arjona Aroca and Antonio Fernandez Anta. Empirical comparison of power-efficient virtual machine assignment algorithms. In *Sustainable Internet and ICT for Sustainability (SustainIT), 2015*, pages 1–8. IEEE, 2015.

[15] Jordi Arjona Aroca, Antonio Fernández Anta, Miguel A. Mosteiro, Christopher Thraves, and Lin Wang. Power-efficient assignment of virtual machines to physical machines. *Future Generation Computer Systems*, 2015. In press, available at http://dx.doi.org/10.1016/j.future.2015.01.006.

[16] Christian Baun, Marcel Kunze, and Viktor Mauch. The koala cloud manager: Cloud service management the easy way. In *Cloud Computing (CLOUD), 2011 IEEE International Conference on*, pages 744–745. IEEE, 2011.

[17] Anton Beloglazov, Jemal Abawajy, and Rajkumar Buyya. Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. *Future Generation Computer Systems*, 28(5):755–768, 2012.

[18] Anton Beloglazov and Rajkumar Buyya. Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers. *Concurrency and Computation: Practice and Experience (CCPE)*, 24(13):1397–1420, 2012.

[19] Anton Beloglazov and Rajkumar Buyya. Openstack neat: A framework for dynamic and energy-efficient consolidation of virtual machines in openstack clouds. *Concurrency and Computation: Practice and Experience (CCPE)*, 27(5):1310–1333, 2015.

[20] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004.

[21] Alaa Abdulkareem Hameed Brihi. Investigation into the dependency between resource utilization, power consumption and performance in multimedia servers.

[22] Eucalyptus. Hpe helion eucalyptus. http://www8.hp.com/us/en/cloud/helion-eucalyptus-overview.html. Accessed January 23th, 2013.

[23] Md Hasanul Ferdaus, Manzur Murshed, Rodrigo N Calheiros, and Rajkumar Buyya. Virtual machine consolidation in cloud data centers using aco metaheuristic. In *Euro-Par 2014 Parallel Processing*, pages 306–317. Springer, 2014.

[24] Joseph L. Hellerstein. Google cluster data. Google research blog, January 2010. Posted at http://googleresearch.blogspot.com/2010/01/google-cluster-data.html.

[25] R. Jansen and P.R. Brenner. Energy efficient virtual machine allocation in the cloud. In *Green Computing Conference and Workshops (IGCC), 2011 International*, pages 1–8, 2011.

[26] Atefeh Khosravi, Saurabh Kumar Garg, and Rajkumar Buyya. Energy and carbon-efficient placement of virtual machines in distributed cloud data centers. In *Euro-Par 2013 Parallel Processing*, pages 317–328. Springer, 2013.

[27] K. Mills, J. Filliben, and C. Dabrowski. Comparing vm-placement algorithms for on-demand clouds. In *Proceedings of the IEEE Third International Conference on Cloud Computing Technology and Science*, pages 91–98, 2011.

[28] Mayank Mishra and Anirudha Sahoo. On theory of vm placement: Anomalies in existing methodologies and their mitigation using a novel vector based approach. In *IEEE CLOUD*, pages 275–282, 2011.

[29] OpenNebula. Opennebula. http://opennebula.org/. Accessed January 23th, 2013.

[30] Charles Reiss, Alexey Tumanov, Gregory R Ganger, Randy H Katz, and Michael A Kozuch. Heterogeneity and dynamicity of clouds at scale: Google trace analysis. In *Proceedings of the Third ACM Symposium on Cloud Computing*, page 7. ACM, 2012.

[31] Charles Reiss, John Wilkes, and Joseph L. Hellerstein. Google cluster-usage traces: format + schema. Technical report, Google Inc., Mountain View, CA, USA, November 2011. Revised 2012.03.20. Posted at http://code.google.com/p/googleclusterdata/wiki/TraceVersion2.

[32] Shekhar Srikantaiah, Aman Kansal, and Feng Zhao. Energy aware consolidation for cloud computing. In *Proceedings of the 2008 conference on Power aware computing and systems*, volume 10 of *HotPower'08*. USENIX Association, 2008.

[33] Hieu Trong Vu and Soonwook Hwang. A traffic and power-aware algorithm for virtual machine placement in cloud data center. *International Journal of Grid & Distributed Computing*, 7(1):350–355, 2014.

[34] Meng Wang, Xiaoqiao Meng, and Li Zhang. Consolidating virtual machines with dynamic bandwidth demand in data centers. In *IEEE INFOCOM*, pages 71–75, 2011.

[35] John Wilkes. More Google cluster data. Google research blog, November 2011. Posted at http://googleresearch.blogspot.com/2011/11/more-google-cluster-data.html.