An energy-efficient algorithm for virtual machine placement optimization in cloud data centers

Sadoon Azizi[1] · Maz'har Zandsalimi[1] · Dawei Li[2]

Abstract

Cloud providers offer computing services based on user demands using the Infrastructure as a Service (IaaS) service model. In a cloud data center, it is possible that multiple Virtual Machines (VMs) run on a Physical Machine (PM) using virtualization technology. Virtual Machine Placement (VMP) problem is the mapping of virtual machines across multiple physical ones. This process plays a vital role in defining energy consumption and resource usage efficiency in the cloud data center infrastructure. However, providing an efficient solution is not trivial due to difficulties such as machine heterogeneity, multi-dimensional resources, and large scale cloud data centers. In this paper, we propose an efficient heuristic algorithm that focuses on power consumption and resource wastage optimization to solve the aforementioned problem. The proposed algorithm, called *MinPR*, minimizes the total power consumption by reducing the number of active physical machines and prioritizing the power-efficient ones. Also, it reduces resource wastage by maximizing and balancing resource utilization among physical machines. To achieve these goals, we propose a new Resource Usage Factor model that manages virtual machine placement on physical machines using reward and penalty mechanisms. Simulations based on cloud user-customized VMs and Amazon EC2 Instances workloads illustrate that the proposed algorithm outperforms existing approaches. In particular, the proposed algorithm reduces total energy consumption by up to 15% for cloud user-customized VMs and by up to 10% for Amazon EC2 Instances.

Keywords Cloud computing · Infrastructure as a service (IaaS) · Virtual machine placement (VMP) · Optimization · Energy efficiency · Resource utilization

1 Introduction

Cloud computing is a relatively new model to offer users compute, network, storage, platform and application services on demand through Internet infrastructure. Cloud computing has become more and more popular because cloud services can be provided in a scalable and elastic fashion based on user demands. Virtualization enables cloud data centers to share a physical machine's resources among multiple tenants.

The ever-increasing demand for cloud infrastructure leads to noticeable energy consumption and greenhouse gas emissions. Based on Amazon's estimations, 42% of a data center's operational cost is due to its energy consumption [1–3]. High energy consumption is a potential threat for cloud providers as it increases the total cost of ownership [4]. High energy consumption comes from both the amount of computational resources used in cloud infrastructure and their power inefficiencies, and the inefficient usage of these resources [5]. Efficient usage of physical machine resources using virtualization is a promising solution to overcome energy inefficiencies. Specifically, Virtual Machine Placement (VMP) optimization has become a trending topic among researchers in recent years [2–4, 6–23].

✉ Sadoon Azizi
  s.azizi@uok.ac.ir

  Maz'har Zandsalimi
  m47h4r@gmail.com

  Dawei Li
  dawei.li@montclair.edu

1 Department of Computer Engineering and IT, University of Kurdistan, Sanandaj, Iran

2 Department of Computer Science, Montclair State University, Montclair, NJ, USA

VMP implies how Virtual Machines (VMs) are placed on Physical Machines (PMs) in a data center. In fact, this problem is an instance of multi-dimensional vector packing, which has no exact polynomial solution [24]. Two main goals of VMP is to minimize power consumption and reduce resource wastage. However, offering an efficient solution for this problem is not trivial due to the following reasons. First, physical machines and user applications (virtual machines) have heterogeneous characteristics. Second, the multi-dimensional nature of VMP problem increases the complexity as server exhaustion in one of the dimensions such as CPU, results in a server being unable to host any more VMs (although other dimensions such as RAM and bandwidth may still have remaining capacities) [8]. Third, finding the optimal placement among thousands of candidates (in case of large scale data centers that receive lots of requests) will further increase the complexity. Although valuable efforts have been made in order to solve this problem, to our knowledge, non of which consider minimizing the number of active physical machines, server power efficiency, and maximizing and balancing resource efficiency altogether.

In this paper, we propose an efficient heuristic algorithm, *MinPR*, with a focus on minimizing the power consumption and resource wastage in the context of large scale data centers that deal with multi-dimensional and heterogeneous resources. The proposed algorithm minimizes power consumption through reducing the number of active physical machines and utilizing power-efficient ones. It also minimizes resource wastage by increasing utilization and balancing different resources on a physical machine. In order to achieve these goals, we propose a new model named *Resource Usage Factor* that handles VM placement using reward and penalty policies. Moreover, by sorting physical machines based on their power efficiency, it is ensured that more power efficient physical machines are utilized first. We also exploit a replacement phase to further minimize resource wastage. The proposed algorithm is compared with existing competitors such as FFD [25, 26], MBFD [4] and RVMP [20] in terms of the total number of active physical machines, total power consumption, total CPU and RAM utilization, and total resource wastage. Simulation results demonstrate that the proposed algorithm is a promising solution to the virtual machine placement problem in cloud data centers.

The rest of this paper is organized as follows. In Sect. 2, related works are reviewed. In Sect. 3, problem definition and its formulation is introduced. The proposed algorithm and its details are presented in Sect. 4. Section 5 is dedicated to performance evaluation and simulation results. Finally, conclusion and future work are given in Sect. 6.

2 Related work

In recent years, several algorithms are proposed to solve the virtual machine placement problem. In general, these algorithms are classified into four categories [11, 19]: heuristic, meta-heuristic, exact, and approximate. Each of these algorithms assigns the virtual machines to servers with certain goals, including minimizing energy consumption, load balancing, maximizing resource utilization, minimizing cost, and reducing Service-Level Agreement (SLA) violation.

Although exact algorithms such as Constraint Programming (CP) [27, 28], Linear Programming (LP) [29], Integer Linear Programming (ILP) [30], Mixed Integer Linear Programming (MILP) [21], Stochastic Integer Programming (SIP) [31], Pseudo-Boolean Optimization (PBO) [32] and Constraint Satisfaction Problem (CSP) [33] produce the optimal solution, their time complexity is exponential. In [34], several approximate algorithms for 1-dimensional bin packing are surveyed. Authors in [35] proved that it is viable to solve the 1-dimensional bin packing with $1 + \epsilon$ approximation in polynomial time. They also proved that 2-dimensional bin packing algorithms cannot be solved in polynomial time. It is worth mentioning that bin-packing is an NP-hard problem and is a special case of VMP problem. Moreover, VMP algorithms must be able to solve problems with more than thousands of VMs and PMs in a reasonable time. Hence, proposing an exact algorithm or even an efficient approximate algorithm is unlikely to be possible [36]. For this reason, most of the similar previous works used heuristic and/or meta-heuristic algorithms [2–6, 9, 10, 12, 14–17, 20, 22, 25, 26, 28, 37–43]. In the following, we discuss some of these algorithms.

First Fit Decreasing (FFD) [25, 26] sorts the VMs decreasingly based on their resource demand and puts the PMs in a consecutive list. For each VM, it searches the PM list and the first PM that has enough resources to host the VM will be chosen. Best Fit Decreasing (BFD) [44] sorts the VMs decreasingly based on their resources similar to FFD; afterwards each VM is assigned to a PM with the minimum residual resource that could fit the VM. FFD and BFD use such a simple strategy, but they do not consider energy consumption and resource balancing among different resources.

Beloglazov et al. [4] proposed the Modified Best Fit Decreasing (MBFD) algorithm in which all VMs are sorted based on their current CPU utilization and for each VM a server that has the least increase in energy consumption will be designated as its destination. In fact, this method chooses the more power-efficient servers first. Esfandiar-poor et al. [10] proposed Our Modified Best Fit Decreasing

(OBFD) algorithm in which VMs are sorted based on their CPU requirement. Afterwards for each VM a server will be found that results in the least energy increasement of the data center. To this end, the best server is chosen from non-underutilized and non-empty servers. The goal of both these algorithms is to minimize the data center's energy consumption but they do not consider maximizing the resource utilization and balancing the load among different types of resources.

Zhang and Ansari [9] formulated the VM placement problem as a bin packing problem. They were inspired from dominant resource, a resource that has the most demanding value among all the others. Hence, they proposed the Dominant Residual Resource aware FFD (DRR-FFD) and its variations. The idea behind DRR-FFD is that first VMs are clustered based on their dominant resource, then in each cluster, VMs are sorted non-increasingly. Also, PMs are sorted non-increasingly based on their power efficiency. DDR-FFD, similar to FFD, chooses the most demanding VM to be placed on the first PM that can host it. But choosing VM from which of the clusters is based on mechanisms that are different in each of the variations. For instance, DDR-BinFill calculates the server's remaining capacity in each iteration and specifies its remaining capacity vector. Finally, it searches the VM cluster based on the dominant remaining capacity vector. Although these heuristics perform better than single dimensional ones, for certain workloads they lead to inappropriate placements [9].

Recently, Gupta and Amgoth [20] proposed Resource-aware Virtual Machine Placement (RVMP) algorithm whose fundamental goals are minimizing energy consumption (through reducing the number of active PMs) and resource wastage. They devised a new technique called Resource Usage Factor that can lead to balanced resource optimization on active PMs. RVMP does no processing on the VM and PM lists whatsoever. Based on VM resource requirements, the resource usage factor of the each PM is calculated. This factor leads to choosing a specific PM for a VM that improves its resource optimization. Although RVMP performs well in maximizing resource optimization and minimizing resource wastage, it does not improve energy consumption dramatically. The reason behind this is that it does not consider PM energy consumption parameters for choosing one to assign to a VM.

The first Ant Colony System (ACS) approach for solving VM placement problem was proposed by Gao et al. [12]. It aims at minimizing power consumption and resource wastage. To reach these goals, the algorithm finds lots of non-dominated solutions and also ignores the resource balance of active PMs. In [7] the authors proposed a VM placement algorithm based on Ant Colony Optimization (ACO) that tries to reduce power consumption and resource wastage by minimizing the number of active PMs. The main issue with this work is that it is not best suited for heterogeneous data centers as the PMs are assumed to be homogeneous. Recently, in [45] authors proposed an ACS-based approach for dynamic VMP whose goal is minimizing the energy consumption in data centers. To achieve this goal, they designed a new heuristic information in which a PM that consumes the least amount of energy is selected as a candidate PM to host VMs. Although their approach improves the energy consumption of data centers, it suffers from high execution time which is an imporant factor in large-scale data centers. Moreover, resource wastage has not been considered in this work.

In [2, 3, 14], different variations of the genetic algorithm are proposed for the VM placement problem in a data center. Proposed algorithms in [2, 3] place the VMs so that energy consumption is reduced in both servers and the network. In [14], authors modeled the VM placement using the vector packing approach, they tried to reduce power consumption through maximizing the efficiency of resource usage and minimizing the number of active PMs. The main focus of these papers is minimizing power consumption, but they did not consider resource efficiency and minimizing resource wastage as algorithms' goals.

Some studies such as [6, 13] used Particle Swarm Optimization (PSO) in order to find an efficient VMP optimization solution. In [6], authors tried to minimize power consumption through redefining the PSO parameters by adopting an energy-aware fitness first strategy to update particle position and exploiting a two-dimensional particle encoding scheme. [13]'s authors also modified PSO so that it can relocate migrated VMs in order to reduce power consumption and VM migration cost. Both of these approaches used the optimized PSO to minimize the data center's energy consumption. However, they ignored resource efficiency balance and minimizing resource wastage.

Although existing algorithms made a great effort to effectively solve the virtual machine placement problem, none of them consider all of the factors below altogether:

1. Physical machine heterogeneity
2. Multi-dimensional resources
3. Minimizing energy consumption through reducing number of active servers and prioritizing efficient servers when assigning virtual machine
4. Minimizing resource wastage by maximizing efficiency and balance among different resources

In this paper, we propose an efficient algorithm for virtual machine placement in large scale data centers that considers all the aforementioned factors.

3 Problem statement and formulation

In this section, we first introduce virtual machine placement preliminaries and then propose the problem formulation.

3.1 Problem statement

Server virtualization is one of the key technologies in cloud computing systems. This technology makes it possible to place and run multiple virtual machines on a single physical one. Once in a while requests are checked and are formed as virtual machines and then cloud infrastructure resources are assigned to these virtual machines. In a cloud computing environment, there is a pool of physical machine resources with different amounts of capacities. The process of mapping multiple virtual machines to a set of physical machines is known as the virtual machine placement problem. This process plays a vital role in efficient usage of the resources and energy consumption in the cloud infrastructure. However, due to reasons like request and physical machine heterogeneity, multi-dimensionality of resources and large scales, proposing an efficient solution is not a straightforward task. This mapping must be designed to satisfy the main requirements of a data center, such as minimizing the energy consumption and cost along with maximizing the profit.

3.2 Problem formulation

In this subsection, we propose the mathematical formulation of the virtual machine placement problem. For ease of readability, the list of notations used throughout the paper is included in Table 1.

Suppose that $P = \{P_1, P_2, \ldots, P_n\}$ is the set of physical machines in the data center where the capacity of d-dimensional resources can be defined as $\boldsymbol{P_p} = (C_p^1, C_p^2, \ldots, C_p^d)$ for $1 \le p \le n$. Also suppose that the set of virtual machines are expressed as $V = \{V_1, V_2, \ldots, V_m\}$ and V_v's d-dimensional resource demand vector is defined as $\boldsymbol{V_v} = (R_v^1, R_v^2, \ldots, R_v^d)$ for $1 \le v \le m$. In order to demonstrate the mapping between VMs and PMs, we use a placement matrix $\boldsymbol{X_{n \times m}}$ whose elements are defined as follows:

$$x_{p,v} = \begin{cases} 1, & \text{if } V_v \text{ is placed on } P_p \\ 0, & otherwise \end{cases} \tag{1}$$

And for the set of PMs, we define an allocation vector $\boldsymbol{y}$ where each y_p, for $1 \le p \le n$, is 1 if it hosts at least one VM; otherwise the value would be 0. In other words:

$$y_p = \begin{cases} 1, & \sum_{v=1}^{m} x_{p,v} \ge 1 \\ 0, & otherwise \end{cases} \tag{2}$$

We define the normalized utilization vector of P_p's resources in all the demensions as $U_p = (U_p^1, U_p^2, \ldots, U_p^d)$, where

$$U_p^i = \sum_{v=1}^{m} \frac{x_{p,v} \times R_v^i}{C_p^i}. \tag{3}$$

Suppose that the CPU capacity of a physical machine P_p is C_p^{cpu} (in Million Instructions Per Second, denoted by MIPS) and it's maximum power consumption is P_p^{max} (in Watt, denoted by W); then the power efficiency of P_p is defined as follows:

$$\mathcal{P}_p = \frac{C_p^{cpu}}{P_p^{max}}. \tag{4}$$

This parameter indicates how much computational power that a specific physical machine provides for each watt. For instance, HP ProLiant G5's computational power and maximum power consumption are 5320 MIPS and 135W, respectively [46]. Therefore, the power efficiency of this server is equal to 39.41. For HP ProLiant G4, the computational power and maximum power consumption are 3721 MIPS and 117W, respectively; thus, its power efficiency is equal to 31.79 [46]. As a result, HP ProLiant G5 is more power efficient than HP ProLiant G4.

In this paper, we focus on two main resource dimensions, CPU and RAM. In other words, $d = 2$. However, the proposed method can be easily extended to any other number of resource dimensions. In the following, the power consumption and resource wastage of a data center is modeled using mathematical words.

3.2.1 Modeling power consumption

In [47], it is shown that it is possible to accurately model the server power consumption linearly as a scale of CPU utilization. Such scale is also proposed by the authors of [12]. In order to save energy, servers turn off when idle; so they do not consume power. Thus, P_p's power consumption is a function of its CPU utilization as shown below:

$$P_p^{power} = \begin{cases} P_p^{idle} + (P_p^{max} - P_p^{idle}) \times U_p^{cpu}, & \text{if } U_p^{cpu} > 0 \\ 0, & otherwise \end{cases} \tag{5}$$

where P_p^{idle} and P_p^{max} are power consumptions of P_p when it is in idle and full utilization, respectively. $U_p^{cpu} \in [0, 1]$ is the normalized CPU utilization. Based on the above formulas, the total power consumption of physical machines in a data center can be calculated by the equation below:

Table 1 List of notations used in this paper

Notation	Description
P	Set of physical machines; $P = \{P_1, P_2, \cdots, P_{|P|}\}$, where $|P| = n$
V	Set of virtual machines; $V = \{V_1, V_2, \cdots, V_{|V|}\}$, where $|V| = m$
d	Number of resource dimensions
C_p^i	Resource capacity of P_p along the ith dimension
R_v^i	Resource demand of V_v along the ith dimension
$x_{v,p}, y_p$	Binary variables
U_p^i	Normalized resource utilization of P_p along the ith dimension
$\mathcal{P}_p$	Power efficiency of P_p
P_p^{idle}	Power consumption of P_p in idle mode
P_p^{max}	Power consumption of P_p in full utilization mode
P_p^{power}	Total power consumption of P_p
L_p^{cpu}	Normalized remaining CPU usage of P_p
L_p^{ram}	Normalized remaining RAM usage of P_p
$P_p^{wastage}$	Total resource wastage of P_p
$\mathcal{A}_p$	Current utilization state of P_p
$\mathcal{B}(p, v)$	Posterior usage state of P_p after placement of V_v
$\mathcal{F}(p, v)$	Resource usage factor of P_p after placement of V_v
$\mathcal{U}_p^i$	Posterior usage state of P_p along the ith dimension
$\mathcal{V}(\mathcal{U}_p^1, \mathcal{U}_p^2, \ldots, \mathcal{U}_p^d)$	Variance of the posterior utilization state vector of PM P_p
$\alpha, \beta, \gamma, \sigma$	Constant values

$$\sum_{p=1}^{n} P_p^{power}$$
$$= \left(\sum_{p=1}^{n} y_p \times (P_p^{idle} + (P_p^{max} - P_p^{idle}) \times \sum_{v=1}^{m} \frac{x_{p,v}.R_v^{cpu}}{C_p^{cpu}} \right) \tag{6}$$

3.2.2 Modeling resource wastage

The remaining available resources of each physical machine can be variant on the VM placement strategy. To fully utilize multi-dimensional resources (such as CPU and RAM), the following equation is used to compute the resource wastage of P_p [12]:

$$P_p^{wastage} = \frac{|L_p^{cpu} - L_p^{ram}| + \epsilon}{U_p^{cpu} + U_p^{ram}} \tag{7}$$

where U_p^{cpu} and U_p^{ram} show the normalized CPU and RAM resource usage, respectively, and L_p^{cpu} and L_p^{ram} determine the normalized remaining CPU and RAM resource usage. ϵ is a very small positive real number that equals to 0.0001 [12]. The key idea behind this modeling is to make effective use of a PM's resources in all dimensions and balance between remaining resources across multiple

dimensions. Hence, the total resource wastage of a data center can be calculated as follows:

$$\sum_{p=1}^{n} P_p^{wastage} = \sum_{p=1}^{n} y_p$$
$$\times \frac{|(C_p^{cpu} - \sum_{v=1}^{m}(x_{p,v} \times R_v^{cpu})) - (C_p^{ram} - \sum_{v=1}^{m}(x_{p,v} \times R_v^{ram}))| + \epsilon}{\sum_{v=1}^{m}(x_{p,v} \times R_v^{cpu}) + \sum_{v=1}^{m}(x_{p,v} \times R_v^{ram})} \tag{8}$$

3.2.3 Optimization formula

Our goal is to solve the optimization problem of VM placement in a data center in a way that both power consumption and resource wastage are minimized. So the problem can be formulated as below:

$$\text{Minimize} \sum_{p=1}^{n} P_p^{power} \tag{9}$$

$$\text{Minimize} \sum_{p=1}^{n} P_p^{wastage} \tag{10}$$

Subject to the following constraints:

$$\sum_{p=1}^{n} x_{p,v} = 1 \quad \forall v = 1, 2, \cdots, m. \tag{11}$$

$$\sum_{v=1}^{m} (x_{p,v} \times R_v^{cpu}) \leq (C_p^{cpu} \times y_p) \quad \forall p = 1, 2, \cdots, n. \tag{12}$$

$$\sum_{v=1}^{m} (x_{p,v} \times R_v^{ram}) \leq (C_p^{ram} \times y_p) \quad \forall p = 1, 2, \cdots, n. \tag{13}$$

$$y_p, x_{p,v} \in \{0, 1\} \quad \forall p = 1, 2, \cdots, n \text{ and } \forall v = 1, 2, \cdots, m. \tag{14}$$

Based on constraint (11), each VM can be hosted only on one PM. Constraints (12) and (13) indicate that the sum of CPU and RAM of all virtual machines hosted on a PM must not exceed the amount of the PM's CPU and RAM capacity, respectively. At last, constraint (14) specifies the problem's variable domains. Having m virtual machines and n physical ones, there are n^m solutions for the placement problem in total. Therefore, proposing an exact solution is infeasible in large scale domains. In the following, we propose an efficient heuristic algorithm that is able to find a good solution in polynomial time for this problem.

4 Proposed algorithm

In this section, we first describe the multi-dimensional resource usage model and then present the proposed algorithm for VM placement.

4.1 Multi-dimensional resource usage model

Based on our problem formulation, each physical machine has d dimensions of resources. To minimize resource wastage in a data center, it is necessary that for each active physical machine, resource utilization in all dimensions are maximized and balanced. To illustrate the PM's d-dimensional utilization we exploit the multi-dimensional resource usage model [20]. A similar approach titled "multi-dimensional space partition model" is previously introduced in [37]. Using this model, it is possible to measure the appropriate resource utilization of each PM for VM placement.

We use the normalized utilization vector to represent the current utilization state of the active physical machine P_p which is represented as $\mathcal{A}_p = (U_p^1, U_p^2, \ldots, U_p^d)$ where U_p^i is calculated based on Eq. (3). The current utilization state of a physical machine can be represented by a point in a d-dimensional domain. For instance, Fig. 1 illustrates the current utilization state of P_p with CPU utilization equal to 50% and RAM utilization equal to 25%, i.e.
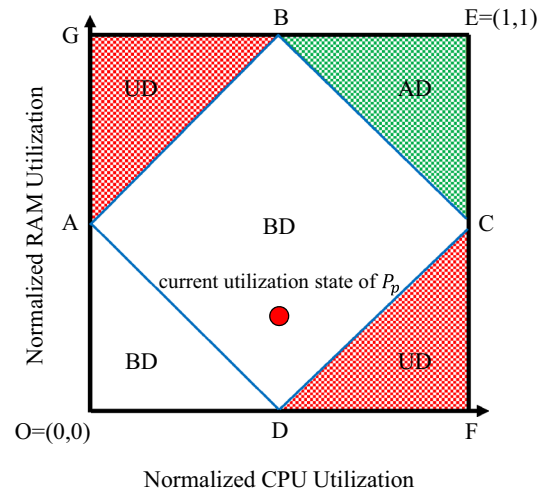

Fig. 1 Multi-dimensional resource usage model

$\mathcal{A}_p = (0.50, 0.25)$. In this figure, point $O = (0, 0)$ represents the idle state of a machine whilst point $E = (1, 1)$ corresponds to the state where P_p is fully utilized in both dimensions. Based on this model, three different domains can be defined as follows [20]:

Acceptance Domain (AD): This domain specifies that d dimensions of a physical machine are balanced properly and that the resource wastage is low. This is an acceptable state for active physical machines.

Balanced Domain (BD): This domain includes those PMs which do not have significant imbalance among their different resources and that there is relative balance overall. However, there are parts of this domain that must be de-prioritized using different mechanisms. We considered this phenomenon in our proposed algorithm (by applying penalty using the variance of the normalized utilization vector).

Unbalanced Domain (UD): This domain specifies that there is a disequilibrium among resource utilizations in different dimensions and that it may cause extreme resource wastage on a physical machine. In a VM placement problem, this must be avoided. We also considered this situation in our proposed algorithm (we de-prioritize this domain by applying some penalty).

In general, the domain determination function $f(U_p)$ for a normalized utilization vector with $U_p = (U_p^1, U_p^2)$ is defined as follows:

$$f(U_p) = \begin{cases} AD, & \text{if } U_p \text{ belong to the triangle BEC} \\ UD, & \text{if } U_p \text{ belong to the triangle AGB or CFD} \\ BD, & \text{otherwise} \end{cases} \tag{15}$$

Consider a $P_p = (C_p^1, C_p^2, \ldots, C_p^d)$ and a $V_v = (R_v^1, R_v^2, \ldots, R_v^d)$, and also suppose that P_p has enough resources to host V_v. Now suppose that the current

utilization state of P_p equals to $\mathcal{A}_p = (U_p^1, U_p^2, \ldots, U_p^d)$. The posterior usage state is P_p's new utilization state after it hosts V_v and we denoted it by $\mathcal{B}(p, v) = (\mathcal{U}_p^1, \mathcal{U}_p^2, \ldots, \mathcal{U}_p^d)$ where $\mathcal{U}_p^i = U_p^i + \dfrac{R_v^i}{C_p^i}$ for $1 \le i \le d$. The posterior utilization state demonstrates the appropriateness of placement of V_v on P_p. For instance, consider an HP ProLiant G4 with specification as $P_p = (3720 \text{ MIPS}, 4 \text{ GB})$ along with the current utilization state $\mathcal{A}_p = (0.5, 0.25)$. Assume that there are two VM requests $V_1 = (1600 \text{ MIPS}, 800 \text{ MB})$ and $V_2 = (1100 \text{ MIPS}, 2200 \text{ MB})$ to be placed on P_p. By placing V_1 on P_p, the utilization state becomes $\mathcal{B}(p, 1) = (0.93, 0.45)$. However, by placing V_2 on P_p, the utilization state $\mathcal{B}(p, 2) = (0.80, 0.79)$ is obtained (see Fig. 2).

4.2 The proposed algorithm for virtual machine placement

The main idea of our algorithm lies in two phases. In the first phase, called *placement phase*, physical machines are sorted in non-ascending order based on their power efficiency as shown in Eq. (4). This means that in a data center with heterogeneous physical machines, those which add the least power increase to data center's power consumption are chosen first to host virtual machines. It is worth noting that if a physical machine's remaining capacity crosses a certain threshold (either gets higher than a certain upper threshold or lower than a bottom one), it will be omitted from the list of available physical machines. In other words, over-loaded and under-loaded machines are not

chosen for hosting. This helps us reduce the data center's power consumption by migrating VMs on under-loaded physical machines so that we can turn that PM off. After choosing a power-efficient physical machine, the algorithm tries to optimize its resource utilization and balance its resource usage across multiple dimensions so that the number of active physical machines and resource wastage is minimized. Details of this phase are discussed in the following subsection. The second phase is the *replacement phase*. If a physical machine P_p's normalized utilization vector resides in the Unbalanced Domain, an inactive PM from the list of all inactive ones is found, such that 1.) it is more power-efficient than the others, 2.) it has enough capacity to host all virtual machines that placed on P_p, and 3.) its utilization vector falls in either the Acceptance Domain (first priority) or Balanced Domain (second priority). Afterwards, P_p will be turned off, and the newly chosen PM will be turned on.

4.2.1 Placement phase

Suppose that there are m virtual machines in queue waiting to be placed and P_p is the most power efficient one, where $\mathcal{A}_p = (U_p^1, U_p^2, \ldots, U_p^d)$. First, we calculate the posterior utilization state of P_p for all VMs like V_v that can be hosted on it (i.e., $\mathcal{B}(p, v) = (\mathcal{U}_p^1, \mathcal{U}_p^2, \ldots, \mathcal{U}_p^d)$). To decide which VM is appropriate to be placed on P_p, we exploit a criterion named *Resource Usage Factor*. This criterion corresponds to the appropriateness of placing V_v on P_p; it can be calculated by the following formula:

$$\mathcal{F}(p, v) = \prod_{i=1}^{d} \mathcal{U}_p^i + f_1(\mathcal{B}(p, v)) - (f_2(\mathcal{B}(p, v)) + f_3(\mathcal{B}(p, v)))$$

$$(16)$$

where $f_1(\mathcal{B}(p, v))$ is a reward function and $f_2(\mathcal{B}(p, v))$ and $f_3(\mathcal{B}(p, v))$ are penalty functions, which can be calculated by using the following relationships:

$$f_1(\mathcal{B}(p, v)) = \begin{cases} \alpha * \prod_{i=1}^{d} \mathcal{U}_p^i, & \text{if point } \mathcal{B}(p, v) \text{ belongs to AD} \\ 0, & \text{otherwise} \end{cases}$$

$$(17)$$

$$f_2(\mathcal{B}(p, v)) = \begin{cases} \beta * \prod_{i=1}^{d} \mathcal{U}_p^i, & \text{if point } \mathcal{B}(p, v) \text{ belongs to UD} \\ 0, & \text{otherwise} \end{cases}$$

$$(18)$$
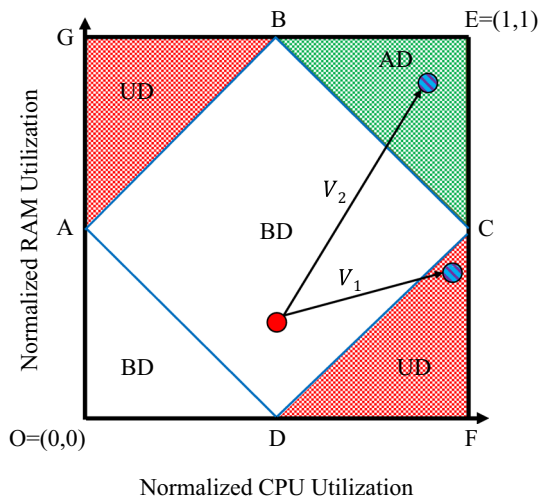
Fig. 2 Posterior Utilization State of $P_p = (3720 \text{ MIPS}, 4 \text{ GB})$ after placing $V_1 = (1600 \text{ MIPS}, 800 \text{ MB})$ or $V_2 = (1100 \text{ MIPS}, 2200 \text{ MB})$

$$f_3(\mathcal{B}(p,v)) = \begin{cases} \gamma * \mathcal{V}(\mathcal{U}_p^1, \mathcal{U}_p^2, \ldots, \mathcal{U}_p^d), & \text{if point } \mathcal{B}(p,v) \text{ in at least one dimension is greater than } \sigma\% \\ 0, & \text{otherwise} \end{cases} \qquad (19)$$

in which $\mathcal{V}(\mathcal{U}_p^1, \mathcal{U}_p^2, \ldots, \mathcal{U}_p^d)$ is the variance of the posterior utlization state vector of PM P_p and α, β, γ and σ are adjustable parameters. In the following, we elaborate each term of Eq. (16) in detail.

The first term (i.e., $\prod_{i=1}^d \mathcal{U}_p^i$) allows us to give a higher priority to a VM which leads to the higher resource utilization in a more balanced way. For example, if we have three VMs V_1, V_2 and V_3, where for each of them we have $\mathcal{B}(p,1) = (0.70, 0.20)$, $\mathcal{B}(p,2) = (0.50, 0.30)$ and $\mathcal{B}(p,3) = (0.30, 0.20)$, respectively. In this case V_2 gets a higher priority to be placed on PM P_p. The second term rewards VM V_v that brings PM P_p to the Acceptance Domain, while the third term penalises a VM that causes PM P_p falls into the Unbalanced Domain. Finally, the fourth term, plays an important role in our placement phase. It is likely that the first and second terms give a high value to a VM for placing on a PM while there exist better choices. Let us explain this case with the help of an illustrative example. Consider two VMs V_1 and V_2, and the posterior utlization state vectors $\mathcal{B}(p,1) = (0.75, 0.60)$ and $\mathcal{B}(p,2) = (0.95, 0.55)$ for PM P_p, respectively. Here both terms one and two prefer V_2 to V_1, while term three has no role on this scenario. However, it is clear that by hosting V_2

on P_p, a lot of resources of PM P_p is wasted. Thus, we prevent this by using an intelligent strategy. For this end, we employ the variance of the posterior utlization state vector of PM P_p. In this example, the variance of VM V_1 is 0.005625 while it is 0.04 for VM V_2. Therefore, our fourth term will impose a lot of penalty to VM V_2.

The algorithm's pseudo code is presented in Algorithm 1. In the proposed algorithm, we assume that all requested VMs can be hosted on the available PMs. Here is how the algorithm performs. First, available physical machines are sorted based on their power efficiency (line 1). Then the first PM, say P_p, is chosen and so we set the number of active physical machines to one (lines 2 and 3). Afterwards, until all the VMs are placed, for each $V_v \in V$ if that VM can be hosted on P_p, the $\mathcal{F}(p,v)$ is calculated for it (lines 4 to 8). At last, a VM that has the largest $\mathcal{F}(p,v)$ is chosen (lines 9 to 11). Then the selected VM called V_{index} is placed on P_p and removed from V; also, P_p's resources are updated (lines 12 to 15). If P_p does not have the capacity to host any more VM, the next P_p is chosen from P and the number of active physical machines is incremented (lines 16 to 18). At last, to balance the resource utilization, the replacement phase is called in line 19.

Algorithm 1 Placement algorithm

Input: P, V

Output: placement of VMs, number of active PMs

1: Sort P in the non-ascending order of their power efficiency.
2: Activate the first P_p from the P
3: $count \leftarrow 1$
4: **while** $m > 0$ **do**
5:     $max \leftarrow 0$
6:     **for all** V_v in V **do**
7:         **if** $R_v^{cpu} \leq C_v^{cpu}$ && $R_v^{ram} \leq C_v^{ram}$ **then**
8:             Calculate $\mathcal{F}(p,v)$
9:             **if** $max < \mathcal{F}(p,v)$ **then**
10:                 $max \leftarrow \mathcal{F}(p,v)$
11:                 $index \leftarrow v$
12:     **if** $max \neq 0$ **then**
13:         $C_p^{cpu} \leftarrow C_p^{cpu} - R_{index}^{cpu}$
14:         $C_p^{ram} \leftarrow C_p^{ram} - R_{index}^{ram}$
15:         Delete V_{index} from V and m--
16:     **else**
17:         Activate the next P_p from the P
18:         count++
19: Call VM replacement phase

4.2.2 Replacement phase

The replacement phase's pseudo code is presented in Algorithm 2. In this phase, first P is divided into two parts: powered-on PMs (P_{on}) and powered-off PMs (P_{off}). Then the list of powered-off PMs are sorted in a non-ascending order based on their power efficiency (lines 1 and 2). Afterwards, in lines 3 and 4, for each powered-on PM the current utilization state is calculated. If the current utilization state of P_p falls into Unbalanced Domain, from powered-off PMs one is chosen (say P_q) that is able to host all VMs on P_p and also will fall into the Acceptance Domain after the placement (lines 5 to 10). Otherwise, in lines 11 to 13, a PM is chosen that has enough resources to host all the VMs on P_p and that after the placement its utilization state falls into Balanced Domain.

Hence, if the number of VMs is larger than the number of PMs, the complexity will be $O(m^2)$; if the number of PMs is much larger than that of VMs ($n >> m$), the time complexity becomes $O(n \log n)$. The second phase's time complexity is quite simple and is, in the worst case, $O(n^2)$.

5 Performance evaluation

In this section, we first define performance metrics. Then we describe the experimental setup and finally present the simulation results. We compare our proposed algorithm, MinPR, to various existing VM placement algorithms: FFD [26], MBFD [4] and RVMP [20].

Algorithm 2 Replacement algorithm

    Input: P, V
    Output: replacement of VMs
1: Classify P into two categories: power on PMs (P_{on}) and power off PMs (P_{off})
2: Sort P_{off} list in the non-ascending order of their power efficiency
3: **for all** P_p in P_{on} **do**
4:     Compute $\mathcal{A}_p$
5:     **if** $\mathcal{A}_p \in UD$ **then**
6:         **for all** P_q in P_{off} **do**
7:             **if** all VMs on P_p can be placed on P_q **&&** after replacement $\mathcal{A}_q \in AD$ **then**
8:                 Remove all VMs from P_p and put them on P_q
9:                 Update the P_{on} and P_{off} list
10:                 **break**
11:             **if** all VMs on P_p can be placed on P_q **&&** after replacement $\mathcal{A}_q \in BD$ **then**
12:                 Remove all VMs from P_p and put them on P_q
13:                 Update the P_{on} and P_{off} list

4.2.3 Complexity analysis

Here we discuss the time complexity of our proposed algorithm. For the placement phase, line 1's complexity is $O(n \log n)$. The while loop continues until all the VMs are placed. On the other hand, the inner for loop is also run for all the VMs that are not placed yet. As a result, the time complexity of these two loops is equal to $O(m^2)$. Since the V is not sorted, the deletion can be done in constant time, by swapping the VM which we want to delete with the last VM and decrementing the array size by one. Overall, the first phase's time complexity is equal to $O(n \log n + m^2)$.

5.1 Performance metrics

In order to compare the algorithms' performances, we used multiple metrics that are defined below.

Number of active PMs: This metric represents the total number of needed physical machines to host VMs.

Total power consumption: We use this metric to compare the total power consumption of PMs in a data center, according to the presented formula in Eq. (6).

Total resource wastage: The total resource wastage metric specifies how the algorithms use the resources of PMs along different dimensions (Eq. 8).

Overall CPU and RAM utilizations: These metrics correspond to the mean values of the CPU and RAM utilizations of k active PMs, and they can be calculated by the equations below:

$$C^{overall} = \frac{1}{k} \sum_{p=1}^{k} \left(\sum_{v=1}^{m} \frac{x_{p,v} \times R_v^{cpu}}{C_p^{cpu}} \right) \quad (20)$$

$$R^{overall} = \frac{1}{k} \sum_{p=1}^{k} \left(\sum_{v=1}^{m} \frac{x_{p,v} \times R_v^{ram}}{C_p^{ram}} \right) \quad (21)$$

5.2 Experiment setup

To conduct experimental simulations, C++ programming language is used. For the proposed algorithm, α and β parameters are set to 0.25 and 0.5, respectively, and γ and σ are set to 4 and 70, respectively. For physical machines, we assume server types: HP ProLiant G4 (3720 MIPS, 4 GB) and HP ProLaint G5 (5320 MIPS, 4 GB). G4's power consumption in the idle state is 86 W, and in full CPU utilization is 117 W. For G5 these values are 93.7 W and 135 W, respectively [46].

For virtual machines, similar to [20], we have considered two scenarios: cloud user-customized VMs and Amazon EC2 Instances. For the first one, we have considered CPU request bounds from 500 to 2500 MIPS and RAM request bounds from 500 to 2000 MB. For the latter, four different scenarios are assumed: Micro, Small, Extra Large and High-CPU Medium instances whose characteristics are shown in Table 2.

5.3 Simulation results

In this subsection, we first propose the simulation results for cloud user-customized VMs (Fig. 3a–f) and then for Amazon EC2 Instances (Fig. 4a–f). In all the tests, the horizontal axis represents the number of virtual machines and their domain varies from 200 to 1000. The vertical axis represents the measured metric in each case. It is worth mentioning that algorithms MBFD and RVMP suppose that all PMs are active initially and the idle one are powered off afterwards, whilst our proposed algorithm and algorithm FFD suppose that all the PMs are powered off by default and are turned on when needed.

5.3.1 Simulation results for cloud user-customized VMs

Figure 3a shows the number of active physical machines against the number of virtual ones. It is observed from the figure that the proposed algorithm MinPR needs much less physical machines to host all the virtual machines. This improvement is more significant when the number of VMs

increases. The number of used PMs in RVMP is less than FFD but is more than the proposed algorithm and MBFD. Analyzing the behaviors of algorithms helped us realize that RVMP prefers G4 servers to G5 ones but our algorithm and MBFD are acting the opposite. Hence, due to the more CPU resources of G5 servers, they are able to activate less PMs leading to our algorithm and then MBFD to perform better. In Fig. 3b different algorithms are compared to one another based on their power consumption. Based on the results, power consumption in a data center is significantly improved when using our proposed algorithm. For instance, for 400 VMs, the proposed algorithm outperforms by approximately 13%, 9% and 11% compared to FFD, MBFD and RVMP, respectively. Results of resource wastage are depicted in Fig. 3c. It is obvious from the figure that the proposed algorithm and then RVMP waste less resources when compared to MBFD and FFD. This is due to that PM resources are utilized in a balanced manner in the proposed algorithm and RVMP. Figure 3d, e illustrate the overall RAM and CPU utilizations, respectively. It is observable from these figures that the proposed algorithm handles resource utilization and balance well. Finally, we have conducted an experiment to observe the number of active PMs for the case where the servers are homogeneous. In order to do so, we assume that all servers are HP ProLiant G5 (5320 MIPS, 4 GB) ones. As depicted in Fig. 3f, the proposed algorithm performs better in this experiment too as it leads to a fewer number of active PMs.

5.3.2 Simulation results for Amazon EC2 Instances

Based on VM specifications these simulation results are somewhat different compared to the previous section. Here we discuss these differences. The number of active PMs that each algorithm produces when trying to host Amazon EC2 Instances are depicted in Fig. 4a. As the figure shows, the proposed algorithm again uses fewer PMs to host Amazon EC2 Instances. However this difference is less significant here. This is due to the VM requirements in this scenario. Specifically, High-CPU Medium and Extra Large Instances reduce the number of options for the algorithms. However, it is important that which PMs host the VMs. Here, in contrast to the previous section, RVMP performs better than MBFD since it places VM so that PMs are

Table 2 VM characteristics for Amazon EC2 Instances scenario

| Instance	CPU (in MIPS)	RAM (in MB)
Micro	500	613
Small	1000	1700
Extra large	2000	3750
High-CPU medium	2500	850
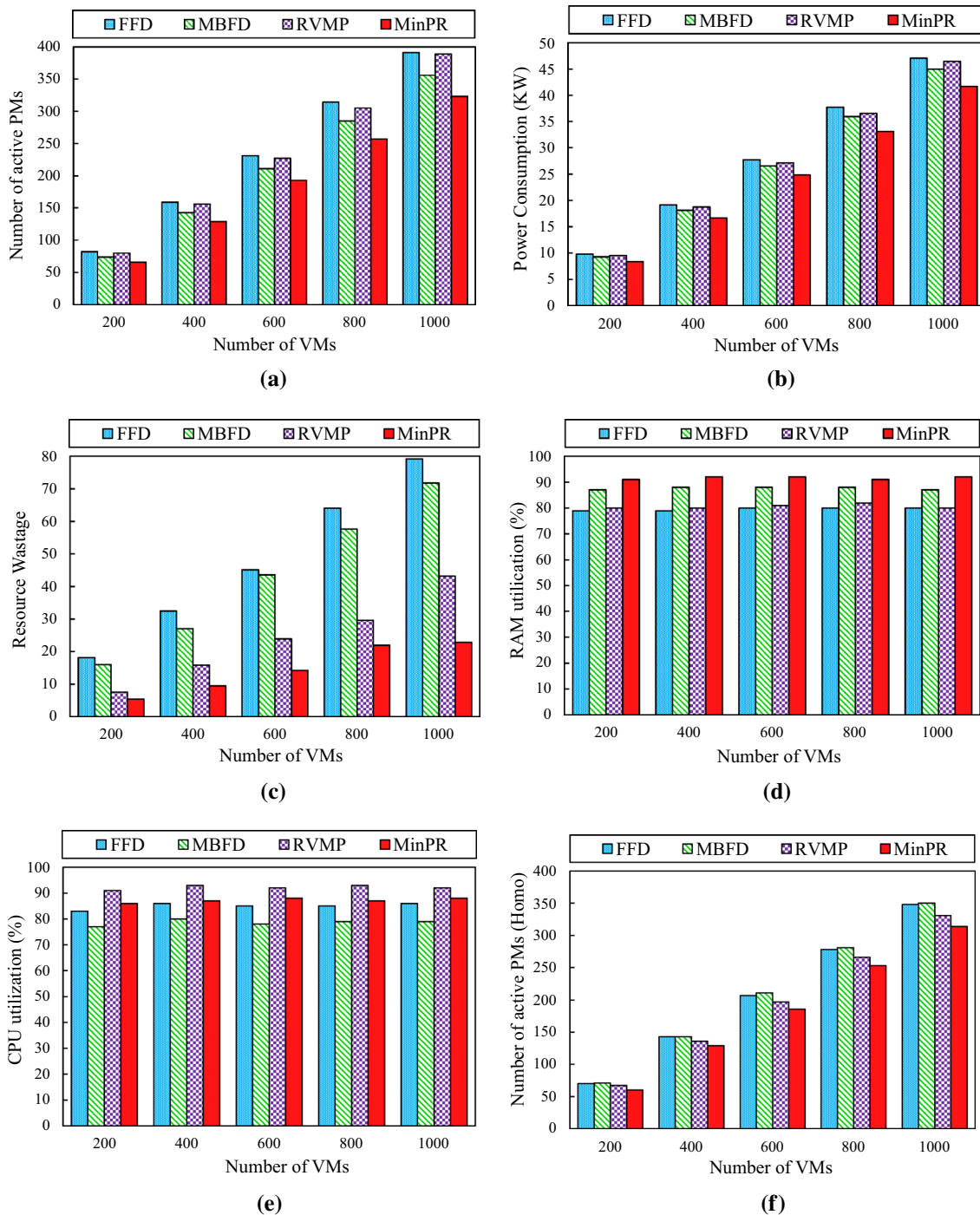
Fig. 3 Simulation results for cloud user-customized VMs. *Homo* homogeneous PMs

utilized properly. The power consumption values of different algorithms are shown in Fig. 4b. By placing VMs on more power efficient PMs and also by using the replacement phase, the proposed algorithm is the most energy efficient one among the four algorithms. The reason why MBFD is the least efficient in power consumption here is that this algorithm hosts most VMs on G5 servers; however, the requirement of VMs in this scenario is somehow

that if the balance between resources is not satisfied, it results in resource wastage and thus high power consumption. Here the improvement of the proposed algorithm for 400 VMs is about 7.5%, 9.5% and 2.5% over FFD, MBFD and RVMP, respectively.

Results of total resource wastage validation is illustrated in Fig. 4c. It is obvious that the proposed algorithm minimizes the resource wastage in the data center properly.
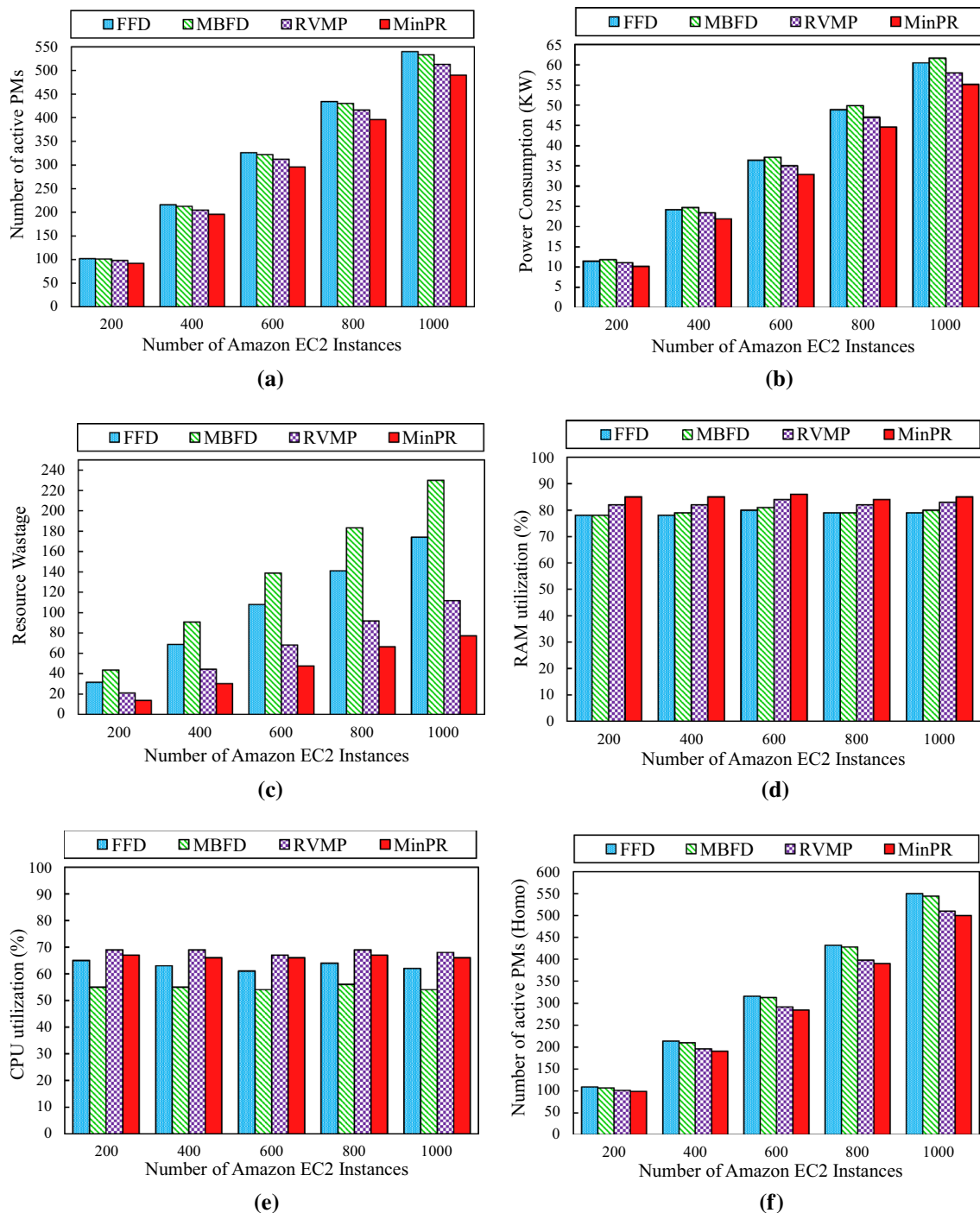
Fig. 4 Simulation results for Amazon EC2 Instances. Homo:= Homogeneous PMs

Although our algorithm places most of the VMs on G5 servers, in contrast to MBFD, it tries to keep the utilization and resource balance at a high ratio. In addition to this, the replacement phase migrates the Extra Large Instances from G5 servers to G4 ones. This leads to the minimization of energy consumption and resource wastage in the data center. It is notable that in Fig. 4d, e the proposed

algorithm offers better utilization and resource balance. We conduct the homogeneous experiment (with all servers being HP ProLiant G5 ones) for this scenario too. Figure 4f shows that in this scenario, our algorithm still performs better compared to the other aforementioned algorithms.

6 Conclusion and future work

Due to the heterogeneous nature of physical and virtual machines, multi-dimensional resources and large scales of cloud data centers, virtual machines placement has become an important research topic. In this work, we proposed an efficient heuristic algorithm to solve this problem which considers power consumption and resource wastage minimization as its goals. The proposed algorithm achieved power consumption minimization by reducing the number of active physical machines and prioritizing the power-efficient ones; it also minimized resource wastage by balancing different resources and improving the utilization of active physical machines. We compared the proposed algorithm with various existing ones through extensive simulations using different perfromance metrics. Results showed that the proposed algorithm significantly reduces the total power consumption and resource wastage of a data center compared to other algorithms. Specifically, compared to the second-best algorithm, our algorithm demonstrated 8% to 12% in the total power efficiency for cloud user-customized VMs and 5% to 8% for the Amazon EC2 Instances. In regard to the total resource wastage, the percentage of improvement was between 26 and 47% for cloud user-customized VMs, while it was about 28% to 35% for the Amazon EC2 Instances. As a future work, we would like to consider the dependency between VMs and the data center network topology in our model. Furthermore, we have a plan to compare our proposed algorithm with more related works, especially meta-heuristic algorithms.

References

1. Teng, F., Yu, L., Li, T., Deng, D., Magoulès, F.: Energy efficiency of vm consolidation in iaas clouds. J. Supercomput. **73**(2), 782–809 (2017)
2. Wu, G., Tang, M., Tian, Y.C., Li, W.: Energy-efficient virtual machine placement in data centers by genetic algorithm. In: International Conference on Neural Information Processing, pp. 315–323. Springer (2012)
3. Tang, M., Pan, S.: A hybrid genetic algorithm for the energy-efficient virtual machine placement problem in data centers. Neural Process. Lett. **41**(2), 211–221 (2015)
4. Beloglazov, A., Abawajy, J., Buyya, R.: Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. Fut. Gener. Comput. Syst. **28**(5), 755–768 (2012)
5. Ferdaus, M.H., Murshed, M., Calheiros, R.N., Buyya, R.: Multi-objective, decentralized dynamic virtual machine consolidation using aco metaheuristic in computing clouds. arXiv preprint arXiv:1706.06646 (2017)
6. Wang, S., Liu, Z., Zheng, Z., Sun, Q., Yang, F.: Particle swarm optimization for energy-aware virtual machine placement optimization in virtualized data centers. In: Parallel and Distributed Systems (ICPADS) pp. 102–109 (2013)
7. Ferdaus, M.H., Murshed, M., Calheiros, R.N., Buyya, R.: Virtual machine consolidation in cloud data centers using aco metaheuristic. In: European Conference on Parallel Processing, pp. 306–317. Springer (2014)
8. Mishra, M., Sahoo, A.: On theory of vm placement: Anomalies in existing methodologies and their mitigation using a novel vector based approach. In: IEEE CLOUD, pp. 275–282. Citeseer (2011)
9. Zhang, Y., Ansari, N.: Heterogeneity aware dominant resource assistant heuristics for virtual machine consolidation. In: Global Communications Conference (GLOBECOM), pp. 1297–1302 (2013)
10. Esfandiarpoor, S., Pahlavan, A., Goudarzi, M.: Structure-aware online virtual machine consolidation for datacenter energy improvement in cloud computing. Comput. Electr. Eng. **42**, 74–89 (2015)
11. Pires, F.L., Barán, B.: A virtual machine placement taxonomy. In: 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid), pp. 159–168. IEEE (2015)
12. Gao, Y., Guan, H., Qi, Z., Liu, L.: A multi-objective ant colony system algorithm for virtual machine placement in cloud computing. J. Comput. Syst. Sci. **79**(8), 1230–1242 (2013)
13. Dashti, S.E., Rahmani, A.M.: Dynamic vms placement for energy efficiency by pso in cloud computing. J. Exp. Theor. Artif. Intell. **28**(1), 97–112 (2016)
14. Jamali, S., Malektaji, S.: Improving grouping genetic algorithm for virtual machine placement in cloud data centers. In: 4th International Conference on Computer and Knowledge Engineering (ICCKE), pp. 328–333. IEEE (2014)
15. Hieu, N.T., Di Francesco, M., Jääski, A.Y.: A virtual machine placement algorithm for balanced resource utilization in cloud data centers. In: IEEE 7th International Conference on Cloud Computing, pp. 474–481. IEEE (2014)
16. Alboaneen, D.A., Tianfield, H., Zhang, Y.: Metaheuristic approaches to virtual machine placement in cloud computing: a review. In: 15th International Symposium on Parallel and Distributed Computing (ISPDC), pp. 214–221. IEEE (2016)
17. Mann, Z.A., Szabó, M.: Which is the best algorithm for virtual machine placement optimization? Concurr. Comput. **29**(10), e4083 (2017)
18. Baalamurugan, K., Bhanu, S.V.: A multi-objective krill herd algorithm for virtual machine placement in cloud computing. J. Supercomput. (2018)
19. Attaoui, W., Sabir, E.: Multi-criteria virtual machine placement in cloud computing environments: a literature review. arXiv preprint arXiv:1802.05113 (2018)
20. Gupta, M.K., Amgoth, T.: Resource-aware virtual machine placement algorithm for iaas cloud. J. Supercomput. **74**(1), 122–140 (2018)
21. Regaieg, R., Koubaa, M., Osei-Opoku, E., Aguili, T.: Multi-objective mixed integer linear programming model for vm placement to minimize resource wastage in a heterogeneous cloud provider data center. In: 2018 Tenth International Conference on Ubiquitous and Future Networks (ICUFN), pp. 401–406 (2018)
22. Addya, S.K., Turuk, A.K., Sahoo, B., Sarkar, M., Biswash, S.K.: Simulated annealing based vm placement strategy to maximize the profit for cloud service providers. Eng. Sci. Technol. **20**(4), 1249–1259 (2017)
23. Al-Jarrah, O., Al-Zoubi, Z., Jararweh, Y.: Integrated network and hosts energy management for cloud data centers. Trans. Emerg. Telecommun. Technol. **30**(9), e3641 (2019)
24. Chekuri, C., Khanna, S.: On multi-dimensional packing problems. In: Proceedings of the Tenth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 185–194. Society for Industrial and Applied Mathematics (1999)
25. Bobroff, N., Kochut, A., Beaty, K.: Dynamic placement of virtual machines for managing sla violations. In: 10th IFIP/IEEE International Symposium on Integrated Network Management, pp. 119–128. IEEE (2007)

26. Keller, G., Tighe, M., Lutfiyya, H., Bauer, M.: An analysis of first fit heuristics for the virtual machine relocation problem. In: 8th International Conference on Network and Service Management (cnsm) and 2012 Workshop on Systems Virtualiztion Management (svm), pp. 406–413. IEEE (2012)

27. Van, H.N., Tran, F.D., Menaud, J.M.: Autonomic virtual resource management for service hosting platforms. In: ICSE Workshop on Software Engineering Challenges of Cloud Computing, pp. 1–8. IEEE (2009)

28. Lin, W., Zhu, C., Li, J., Liu, B., Lian, H.: Novel algorithms and equivalence optimisation for resource allocation in cloud computing. Int. J. Web Grid Serv. **11**(2), 193–210 (2015)

29. Bellur, U., Rao, C., Madhu Kumar, S.D.: Optimal placement algorithms for virtual machines. arXiv preprint arXiv:1011.5064 (2010)

30. Anand, A., Lakshmi, J., Nandy, S.: Virtual machine placement optimization supporting performance slas. In: 5th International Conference on Cloud Computing Technology and Science, vol. 1, pp. 298–305. IEEE (2013)

31. Chaisiri, S., Lee, B.S., Niyato, D.: Optimal virtual machine placement across multiple cloud providers. Services Computing Conference, 2009. APSCC 2009. IEEE Asia-Pacific, pp. 103–110 (2009)

32. Ribas, B.C., Suguimoto, R.M., Montano, R.A., Silva, F., Castilho, M.: Pbfvmc: a new pseudo-boolean formulation to virtual-machine consolidation. In: Brazilian Conference on Intelligent Systems, pp. 201–206. IEEE (2013)

33. Van, H.N., Tran, F.D., Menaud, J.M.: Performance and power management for cloud infrastructures. In: 3rd international Conference on Cloud Computing, pp. 329–336. IEEE (2010)

34. Coffman, E.G., Csirik, J., Galambos, G., Martello, S., Vigo, D.: Bin Packing Approximation Algorithms: Survey and Classification. Handbook of Combinatorial Optimization, pp. 455–531 (2013)

35. Vega, W.F.D.L., Lueker, G.S.: Bin packing can be solved within $1 + \epsilon$ in linear time. Combinatorica **1**(4), 349–355 (1981)

36. Mann, Z.Á.: Approximability of virtual machine allocation: much harder than bin packing. In: 9th Hungarian-Japanese Symposium on Discrete Mathematics and Its Applications, pp. 21–30 (2015)

37. Li, X., Qian, Z., Lua, S., Wu, J.: Energy efficient virtual machine placement algorithm with balanced and improved resource utilization in a data center. Math. Comput. Model. **58**(5), 1222–1235 (2013)

38. Sun, X., Ansari, N., Wang, R.: Optimizing resource utilization of a data center. IEEE Commun. Surv. Tutor. **18**(4), 2822–2846 (2016)

39. Mollamotalebi, M., Hajireza, S.: Multi-objective dynamic management of virtual machines in cloud environments. J. Cloud Comput. **6**(1), 16 (2017)

40. Abdessamia, F., Zhang, W.Z., Tian, Y.C.: Energy-efficiency virtual machine placement based on binary gravitational search algorithm. Clust. Comput. (2019)

41. Chang, Y., Gu, C., Luo, F., Fu, W.: Energy efficient resource selection and allocation strategy for virtual machine consolidation in cloud datacenters. IEICE Trans. Inf. Syst. **101**(7), 1816–1827 (2018)

42. Satpathy, A., Addya, S.K., Turuk, A.K., Majhi, B., Sahoo, G.: Crow search based virtual machine placement strategy in cloud data centers with live migration. Comput. Electr. Eng. **69**, 334–350 (2018)

43. Masdari, M., Gharehpasha, S., Ghobaei-Arani, M., Ghasemi, V.: Bio-inspired virtual machine placement schemes in cloud computing environment: taxonomy, review, and future research directions. Clust. Comput. (2019)

44. Varasteh, A., Goudarzi, M.: Server consolidation techniques in virtualized data centers: a survey. IEEE Syst. J. **11**(2), 772–783 (2017)

45. Alharbi, F., Tian, Y.C., Tang, M., Zhang, W.Z., Peng, C., Fei, M.: An ant colony system for energy-efficient dynamic virtual machine placement in data centers. Expert Syst. Appl. **120**, 228–238 (2019)

46. Beloglazov, A., Buyya, R.: Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers. Concurr. Comput. **24**(13), 1397–1420 (2012)

47. Fan, X., Weber, D.W., Barroso, L.A.: Power provisioning for a warehouse-sized computer. ACM SIGARCH Comput. Archit. News **35**(2), 13–23 (2007)

Sadoon Azizi received his Ph.D. degree in computer science, with focus on Cloud Data Centers, from Amirkabir University of Technology, Tehran, Iran, in 2016. He also received his M.Sc. degree in computer science, with focus on High Performance Computing (HPC), from Amirkabir University of Technology, Tehran, Iran, in 2012. Currently, he is an assistant professor in the department of computer engineering and IT, University of Kurdistan, Sanandaj, Iran. Since April 2018, he is the director of the High Performance Computing (HPC) center at the University of Kurdistan. His current research interests include cloud computing, fog computing, internet of things, and high performance computing.
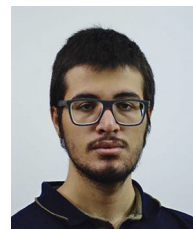
Mazhar Zandsalimi received his B.Sc. degree in computer engineering from University of Kurdistan, Sanandaj, Iran, in 2018. He is currently a researcher at the High Performance Computing (HPC) center at University of Kurdistan. His current research areas include high performance computing, cloud computing and fog computing.

Dawei Li received the bachelor's degree from the department of electronics and information engineering (now school of electronic information and communications), Huazhong University of Science and Technology, Wuhan, Hubei, China, in 2011, and the Ph.D. degree from the department of computer and information sciences, Temple University, in 2016. He is currently an assistant professor with the department of computer science, Montclair State University, Montclair, NJ, USA. His research interests lie in the general fields of parallel and distributed systems, including green computing, data center networks, cloud and edge computing, etc.