GRVMP: A Greedy Randomized Algorithm for Virtual Machine Placement in Cloud Data Centers

Sadoon Azizi, *Member, IEEE*, Mohammad Shojafar, *Senior Member, IEEE*, Jemal Abawajy, *Senior Member, IEEE*, and Rajkumar Buyya

***Abstract*—Cloud computing efficiency greatly depends on the efficiency of the virtual machines (VMs) placement strategy used. However, VM placement has remained one of the major challenging issues in cloud computing mainly because of the heterogeneity in both virtual and physical machines (PMs), the multidimensionality of the resources, and the increasing scale of the cloud data centers (CDCs). An inefficiency in VM placement strategy has a significant influence on the quality of service provided, the amount of energy consumed, and the running costs of the CDCs. To address these issues, in this article, we propose a greedy randomized VM placement (GRVMP) algorithm in a large-scale CDC with heterogeneous and multidimensional resources. GRVMP inspires the "power of two choices" model and places VMs on the more power-efficient PMs to jointly optimize CDC energy usage and resource utilization. The performance of GRVMP is evaluated using synthetic and real-world production scenarios (Amazon EC2) with several performance matrices. The results of the experiment confirm that GRVMP jointly optimizes power usage and the overall wastage of resource utilization. The results also show that GRVMP significantly outperforms the baseline schemes in terms of the performance metrics used.**

***Index Terms*—Cloud computing, cloud data center (CDC), energy and power consumption, greedy randomized algorithm, resource wastage, virtual machine placement (VMP).**

I. Introduction

MODERN cloud data centers (CDCs) [1] hosting cloud computing are very large with large numbers of physical machines (PMs) that are provided to clients as virtual machines (VMs). VMs allow a high level of flexibility in terms of CDC resources management and facilitate the execution of workloads elastically [2] as well as hardware consolidation for maximizing energy efficiency [3]. Moreover, VMs are indispensable in achieving load balancing and fault-tolerance [4] as well as simplifying the sharing of CDC resources among end users efficiently [5].

Although virtualization provides significant benefits in running CDCs, an efficient virtual machine placement (VMP) strategy is required in order to realize fully the many benefits that virtualization provides [6]. A VMP strategy assigns a set of VMs to the most suitable PMs. A VM placement is typically required when new requests for VMs arrive and when the currently running VMs need to be relocated to another PMs. An efficient VM placement strategy maximizes the quality of service offered to end users, minimizes wastage of resource utilization and overall operation costs, reduce network traffic, and substantially reduce CDC consumption of power [7].

As VMP is a critical task in efficiently utilizing CDC resources, it has attracted considerable attention in commercial and the academia [8]–[10]. In fact, VMP is an NP-hard optimization problem [4] for which there are practically no optimal solutions, even for small-scale CDCs. Therefore, many suboptimal solutions have been proposed to address the VM placement problem. Metaheuristic algorithms [11]–[13] usually provide reasonable solutions, but they suffer from high execution time and require parameter tuning. Heuristic algorithms [3], [14], [15] usually have reasonable execution time; however, the quality of their solutions is challenging. Therefore, an efficient solution to the VM placement problem is still an active research topic.

This article addresses the VM placement problem for a heterogeneous, large-scale CDCs. VMP is one of the main challenging CDC problems that has persisted and even exacerbated with the increasing scale of the modern data centers [16]. Even though many solutions to the VM placement problem have been proposed, existing solutions do not consider significant factors arising from the multidimensionality of resources, scale of the modern data centers, and heterogeneity of both VMs and PMs. To this end, we propose an efficient greedy randomized VMP (GRVMP) algorithm with the aim of minimizing the consumption of power and wastage of resource utilization. To achieve these goals, we inspire the "power of two choices" model [17] and combine it with the first fit (FF) greedy heuristic [18]. The proposed approach minimizes the CDC consumption of power by placing VMs on the more power-efficient PMs and maximizing the utilization of the resources. It also reduces the wastages of the resources by balancing the different resources on the activated PMs. In each step, unlike the FF, our approach selects d random choices among the list of VMs and selects the VM that leads to the least resource wastage on the current

Manuscript received January 12, 2020; revised May 6, 2020 and June 8, 2020; accepted June 12, 2020. Date of publication June 30, 2020; date of current version June 7, 2021. This work was supported by an Australian Research Council Discovery Project. The work of Mohammad Shojafar was supported by a Marie Curie Fellowship funded by the European Commission under Agreement MSCA-IF-GF-2019-839255. *(Corresponding author: Sadoon Azizi.)*

Sadoon Azizi is with the Department of Computer Engineering and IT, University of Kurdistan, Sanandaj 66177-15175, Iran (e-mail: s.azizi@uok.ac.ir).

Mohammad Shojafar is with the Institute for Communication Systems/5G Innovation Centre, University of Surrey, Guildford GU27XH, U.K. (e-mail: m.shojafar@surrey.ac.uk).

Jemal Abawajy is with the Faculty of Science, Deakin University, Burwood, VIC 3125, Australia (e-mail: jemal.abawajy@deakin.edu.au).

Rajkumar Buyya is with the Cloud Computing and Distributed Systems Laboratory, School of Computing and Information Systems, University of Melbourne, Melbourne, VIC 3125, Australia (e-mail: rbuyya@unimelb.edu.au).

Digital Object Identifier 10.1109/JSYST.2020.3002721

PM. We conduct an extensive simulation to compare GRVMP with the existing algorithms. The results demonstrate that our algorithm makes a significant improvement in reducing power consumption and resource wastage. Our main contributions are summarized as follows.

1) We formulate the VMP problem as a mixed-integer linear programming (MILP) with the objectives of optimizing consumption of power and resource wastage of a CDC.
2) We propose an efficient greedy randomized algorithm, called *GRVMP*, to solve the VMP problem.
3) We evaluate GRVMP through extensive experiments on both synthetic and Amazon EC2 instances. The results show that GRVMP significantly outperforms the baseline VMP schemes.

The rest of this article is organized as follows. Related works are presented in Section II. Section III highlights the system architecture and problem formulation. The proposed GRVMP is discussed in Section IV. Experimental evaluation of GRVMP and analysis of the results and limitations of our work are discussed in Sections V and VI, respectively. Finally, Section VII provides conclusion and highlights planned future works.

II. Related Work

VM placement strategy is an active research topic and the survey papers in [2], [7], and[19] present the background and the state-of-the-art VM placement strategies. In [20], an algorithm based on simulated annealing for minimizing the consumption of power is proposed. Wang *et al.* [21] introduced an improved particle swarm optimization (PSO) based strategy to reduce the consumption of energy. They improve the PSO through redefining its parameters and operators, plus designing a new 2-D encoding scheme and adopting an energy-aware local fitness first strategy to update the position of particles. Al-Moalmi *et al.* [22] use the gray wolf optimization method to reduce the consumption of power by optimizing the actively used number of PMs. An approach to select energy-efficient PMs for hosting VMs, which is based on an ant colony system (ACS), is discussed in [13]. The main goal of the aforementioned approaches is to reduce the consumption of energy mainly by reducing the number of actively used PMs. However, they ignore the reduction of the resource wastage.

An ACS-based approach to place VMs on PMs with the aim of reducing both the consumption of power and the wastage of resource is proposed in [23]. Similar to this work, Liu *et al.* [24] proposed an ACS strategy with a local search mechanism to ensure the fast convergence of the ACS. In [12], Zheng *et al.* formulate the VM consolidated placement as a biobjective problem, and propose a biogeography-based optimization technique to minimize the consumption of power and the wastage of resources. Recently, Parvizi and Rezvani [25] formulate the VMP problem as a nonlinear convex optimization and present the nondominated sorting genetic algorithm to reduce consumption of power, wastage of resources, and the number of PMs in active state. Also, Abohamama and Hamouda [26] propose a hybrid algorithm based on improved genetic algorithm and multidimensional resource-aware best fit strategy to minimize energy consumption by reducing the number of PMs in active state and minimize resource wastage by balancing the usage of the multidimensional resources. Although, these works take into account resource wastage, in addition to power consumption, the power efficiency of PMs has been ignored. Metaheuristic algorithms usually provide reasonable solutions for many optimization problems. Their execution time, however, is not competitive [6], which is an issue for cloud data center networks (CDCN).

Li *et al.* [27] introduce a multidimensional space partition model to characterize the resource usage state of PMs and use EAGLE algorithm to minimize the consumption of energy by reducing the number of PMs in active state and balancing their multidimensional resource utilization. Gupta and Amgoth [28] present a new VMP algorithm based on the devised resource usage factor model to improve the resource usage of PMs. Their simulation results reveal that the approach can jointly minimize the consumption of power and the wastage of resources in the Infrastructure as a Service (IaaS) cloud. Yao *et al.* [29] propose an approach, which is based on weighted PageRank, to reduce the number of PMs used as well as the resource utilization. Wei *et al.* [6] formulate the VMP problem as MILP to minimize energy consumption. Then, they propose a branch-and-bound-based algorithm and three heuristic algorithms, inspired by the best fit decreasing (BFD) algorithm to address the VMP. Although heuristic algorithms provide low execution time, the quality of their solution usually is a challenging task. Thus, providing an algorithm with both low execution time and high efficiency for the VMP problem in a multidimensional, heterogeneous, and large-scale CDCN is a challenging task. Table I summarizes the literature compared to our method and highlights their limitations.

III. System Architecture and Problem Formulation

This section presents our considered framework and formalizes the VM placement problem.

A. System Architecture

The main building blocks of the considered CDCN are shown in Fig. 1. The three-tier architecture is composed of a set of VMs, m hypervisors, m PMs, different set of switches, and management entities. The PMs are capable of hosting multiple VMs. A Hypervisor manages a set of VMs in a PM. Moreover, the PMs are grouped in Pods. A Pod is composed of multiple top-of-rack (ToR) switches that can connect to each server that is located in a rack. In each Pod, multiple ToR switches can connect to multiple aggregation switches (see the second layered aggregation switches marked by blue circles). Additionally, each aggregation switch connects with multiple switches at the core tier, which is located in the CDCN core component (upper part of the figure). We use a fat-tree topology to capture the connection between the network switches and describe the managing strategy that is applied in the proposed architecture [30]. A centralized network manager located in cloud network (see the upper left part of the figure) manages the allocation manager and the CDCN topology and the networking devices. The allocation manager performs the VM placement to fairly distribute the VMs over the PMs and ensure that a VM receives the required amount of CPU and memory from the PM hypervisor.

TABLE I
COMPARISON OF RELATED WORKS

References	Power Efficiency per Server	Multi-dimensionality	CDCN Power Consumption	CDCN Resource Wastage	Runtime
[21]	○	●	●	○	High
[3]	●	○	●	○	Low
[22]	○	●	●	○	High
[28]	○	●	●	●	Low
[24]	○	●	●	●	High
[14]	○	○	●	○	Low
[12]	○	●	●	●	High
[25]	○	●	●	●	High
[29]	○	●	●	●	low
[23]	○	●	●	○	High
[13]	●	●	●	○	High
[30]	○	●	●	●	High
[26]	○	●	●	●	High
[27]	○	●	●	●	High
[6]	○	●	●	●	Low
GRVMP	●	●	●	●	**Low**

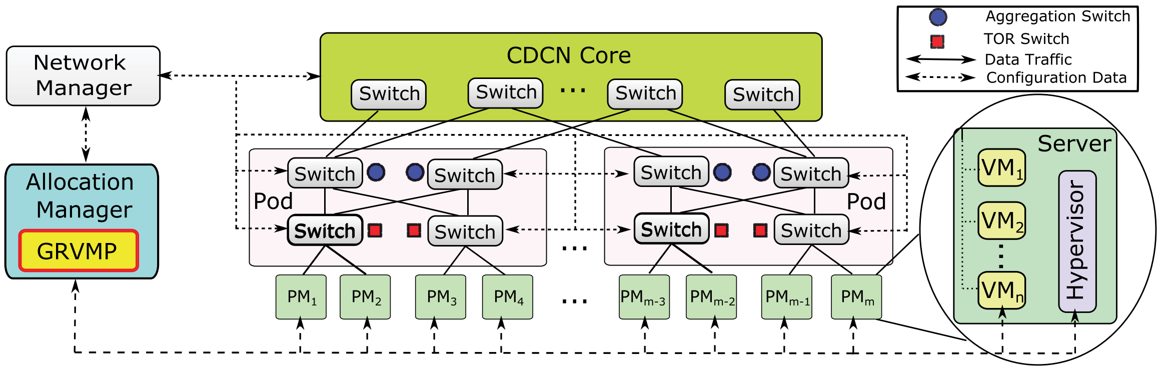

Fig. 1. CDCN architecture. ToR:= top-of-rack switch, PM:= physical machine, VM:= virtual machine, GRVMP:= greedy randomize VM placement algorithm, and $\{a, b, c\}$ are random numbers.

B. VM Placement Problem Formulation

This section presents the mathematical formulation of the VMP problem. Table II presents the main notation used in this article.

1) Physical Machines: Let $P = \{P_1, P_2, \dots, P_m\}$ be a set of m heterogeneous PMs. The capacity of ith PM is represented as a r-dimension vector $P_i = \{C_i^1, C_i^2, \dots, C_i^r\}$, where C_i^k is the kth dimension resource capacity of ith PM, $\forall \boldsymbol{k} = \{1, 2, \dots, r\}$.

2) Virtual Machines: Let $V = \{V_1, V_2, \dots, V_n\}$ be a set of n heterogeneous VMs. The resource requirement of jth VM is defined as a r-dimension vector $V_j = \{R_j^1, R_j^2, \dots, R_j^r\}$, where R_j^k is the kth dimension resource demand of the jth VM, $\forall \boldsymbol{k} = \{1, 2, \dots, r\}$.

3) VMP Problem: VMP is an instance of the multidimensional vector bin packing problem [31]. This problem can be presented as a mapping function $f : \boldsymbol{V} \rightarrow \boldsymbol{P}$ in such a way that some optimization goals and constrained are satisfied. We now formulate the VMP problem as follows.

Let $\boldsymbol{X}_{m\times n}$ be a *binary mapping matrix* to represent the mapping function f between VMs and PMs in which each of its elements is defined in the following equation:

$$x_{ij} = \begin{cases} 1 & \text{if } V_j \text{ is placed on } P_i \\ 0 & \text{otherwise} \quad \forall i \in \boldsymbol{P}, \forall j \in \boldsymbol{V}. \end{cases} \tag{1}$$

We use $\boldsymbol{Y}_n$ as a *binary allocation vector (BAV)* to identify the state of PMs, where each of its element is one when *at least* one VM is hosted on the corresponding PM, otherwise that element would be zero. Hence, we present BAV in the following equation:

$$y_i = \begin{cases} 1 & \text{if } \sum_{j=1}^{n} x_{ij} \geq 1 \\ 0 & \text{otherwise} \quad \forall i \in \boldsymbol{P}. \end{cases} \tag{2}$$

To specify the utilization of a P_i's resources, we define $\mathcal{U}_i^k$ as a *normalized utilization vector*, which is defined in the following equation:

$$\mathcal{U}_i^k = \sum_{j=1}^{n} \frac{x_{ij} \times R_j^k}{C_i^k} \quad \forall i \in \boldsymbol{P}, \forall k \in \boldsymbol{K}. \tag{3}$$

In this article, it is assumed that the allocation manager knows the resource requirements of the VMs and the capacity of the PMs.

Accordingly, the *CPU utilization* of PM P_i, denoted by $\mathcal{U}_i^{\text{cpu}}$, is defined as

$$\mathcal{U}_i^{\text{cpu}} = \sum_{j=1}^{n} \frac{x_{ij} \times R_j^{\text{cpu}}}{C_i^{\text{cpu}}} \quad \forall i \in \boldsymbol{P} \tag{4}$$

where R_j^{cpu} and C_i^{cpu} are, respectively, the CPU demand of VM V_j and the CPU capacity of PM P_i (in million instructions per second [MIPS]).

TABLE II
MAIN NOTATION

	Symbol	Definition	Type - Unit	Appears in Eq.
Set	$\boldsymbol{P}$	Set of PMs, where $\mid \boldsymbol{P} \mid = m$	-	-
	$\boldsymbol{V}$	Set of of VMs, where $\mid \boldsymbol{V} \mid = n$	-	-
	$\boldsymbol{K}$	Set of resource demand dimensions, where $\mid \boldsymbol{K} \mid = r$	-	-
Index	i	Index of PM, $i \in \boldsymbol{P}$	Integer - [units]	-
	j	Index of VM, $j \in \boldsymbol{V}$	Integer - [units]	-
	k	Index of resource dimension, $k \in \boldsymbol{K}$	Integer - [units]	-
	z	Number of active PMs	Integer - [units]	(18), (19)
Input Parameters	C_i^k	Resource capacity of P_i along the k-th dimension	Continuous - [units]	(3), (13)
	R_j^k	Resource demand of V_j along the k-th dimension	Continuous - [units]	(3), (13)
	$\mathcal{U}_i^k$	Normalized resource utilization of P_i along the k-th dimension	Continuous - [units]	(3), (5), (8), (10)
	C_i^{cpu}	CPU capacity of P_i	Continuous - [MIPS]	(18)
	C_i^{mem}	Memory capacity of P_i	Continuous - [MB]	(19)
	R_j^{cpu}	CPU demand of V_j	Continuous - [MIPS]	(18)
	R_j^{mem}	Memory demand of V_j	Continuous - [MB]	(19)
	$\mathcal{U}_i^{cpu}$	Normalized CPU utilization of P_i	Continuous - [units]	(4), (7), (9)
	$\mathcal{P}_i$	Power efficiency of P_i	Continuous - [units]	(6)
	P_i^{min}	Power consumption of P_i in idle mode of each PM	Continuous - [watt]	(7), (9)
	P_i^{max}	Power consumption of P_i in full utilization mode of each PM	Continuous - [watt]	(6), (7), (9)
	P_i^{power}	Power consumption of P_i	Continuous - [watt]	(7), (9)
	$\mathcal{R}_i^k$	Normalized remaining resource of P_i	Continuous - [units]	(5), (8), (10), (12)
	$\mathcal{R}_i^w$	Resource wastage of P_i	Continuous - [units]	(8), (10)
Variables	x_{ij}	Value 1 if V_i placed on P_i, otherwise zero	Binary - [units]	(1), (3), (4), (12),(13)
	y_i	Value 1 if at least one VM is hosted on the corresponding PM, otherwise zero	Binary - [units]	(2), (9), (10)
	$\mathbb{P}^{tot}$	Total power consumption of a CDCN	Continuous - [watt]	(9), (11)
	$\mathbb{R}^{tot}$	Total resource wastage of a CDCN	Continuous - [units]	(10), (11)

We also define $\mathcal{R}_i^k$ to be the *normalized remaining resource vector* of PM P_i, where its element is obtained as follows:

$$\mathcal{R}_i^k = 1 - \mathcal{U}_i^k \quad \forall i \in \boldsymbol{P}, \forall k \in \boldsymbol{K}. \tag{5}$$

In a heterogeneous CDCN, each PM has its own power consumption profile. Let $P_i^{\max}$ denotes the maximum power consumption of PM P_i. Then, we define the power efficiency of this PM as follows [16]:

$$\mathcal{P}_i = \frac{C_i^{\text{cpu}}}{P_i^{\max}} \quad \forall i \in \boldsymbol{P}. \tag{6}$$

Several recent works define various computation models to derive the energy usage of each PM like [32]–[34]. Here, we use a linear power model [3], [35], [36] in which the power consumption of PM P_i is computed as a linear function of its CPU utilization. So, we have

$$P_i^{\text{power}} = \begin{cases} P_i^{\min} + \left(P_i^{\max} - P_i^{\min}\right) \times \mathcal{U}_i^{\text{cpu}} & \text{if } \mathcal{U}_i^{\text{cpu}} > 0 \\ 0 \text{otherwise} & \forall i \in \boldsymbol{P} \end{cases} \tag{7}$$

where $P_i^{\min}$ is the consumption of power by PM P_i when it is in *idle state*.

To determine how a VM placement solution utilizes the multidimensional resources of a PM, we consider the *resource wastage* as well. In literature, the resource wastage formula has been just presented for 2-D resources [23]. Here, we generalized the formula such that it could be applicable for r-dimensional resources. Thus, we use the following equation to calculate the resource wastage of an r-dimensional PM P_i:

$$\mathcal{R}_i^w = \frac{\sum_{k=1}^{r} \mid \mathcal{R}_i^k - \min(\mathcal{R}_i^k) \mid + \epsilon}{\sum_{k=1}^{r} \mathcal{U}_i^k} \quad \forall i \in \boldsymbol{P} \tag{8}$$

where $\min(\mathcal{R}_i^k)$ indicates the minimum normalized remaining resource among all dimensions of the PM P_i. Also, ϵ is a small positive real number that we set its value equals to 0.0001 [23].

Now, we can model the *total power usage* and *resource wastage* of PMs in a CDCN. The total consumption of power by PMs is given as

$$\begin{aligned} \mathbb{P}^{\text{tot}} &= \sum_{i=1}^{m} P_i^{\text{power}} \\ &= \sum_{i=1}^{m} y_i \times \left(P_i^{\min} + \left(P_i^{\max} - P_i^{\min}\right) \times \mathcal{U}_i^{\text{cpu}}\right). \end{aligned} \tag{9}$$

The total resource wastage of PMs in a CDCN is obtained based on the following equation:

$$\mathbb{R}^{\text{tot}} = \sum_{i=1}^{m} \mathcal{R}_i^w = \sum_{i=1}^{m} y_i \times \frac{\sum_{k=1}^{r} \mid \mathcal{R}_i^k - \min(\mathcal{R}_i^k) \mid + \epsilon}{\sum_{k=1}^{r} \mathcal{U}_i^k}. \tag{10}$$

4) Overall Problem Formulation: Our main goal is to solve the VMP problem with the aim of reducing the total power usage and wastage of the resources simultaneously. We formulate the VMP problem as a biobjective problem as follows:

$$\min \left(\mathbb{P}^{\text{tot}} + \mathbb{R}^{\text{tot}}\right) \tag{11}$$

subject to

$$\text{Service constraint:} \sum_{i=1}^{m} x_{ij} = 1 \quad \forall j \in \boldsymbol{V} \tag{12}$$

$$\text{Resource constraint:} \sum_{j=1}^{n} x_{ij} \times R_i^k \leq C_i^k \quad \forall i \in \boldsymbol{P}, \forall k \in \boldsymbol{K} \tag{13}$$

under binary control (decision) variables: $x_{ij} \in \{0, 1\}$, $y_i \in \{0, 1\}$: $i \in \boldsymbol{P} \quad \forall j \in \boldsymbol{V}$.

Constraint (12) ensures that each VM can be assigned to only one PM. Constraint (13) guarantees that the resource demands of all the VMs hosted on any PM should not exceed its corresponding resource capacity.

Indeed, having m PMs and n VMs, there exist m^n solutions for mapping the VMs to the PMs. In a large-scale CDCN, m and n usually are in the scale of thousands [37], [38], which make developing efficient VM placement algorithm a challenging task. In the following section, we address this challenge and propose a greedy randomized algorithm that can find an efficient real-time solution, even for very large scale ms and ns.

Theorem 1: VMP is in the class of NP-hard problems.

Proof: We consider a simplified case of the VMP where the amount of resources for each PM denote by $\tilde{R}_j^k$ is preliminary assigned. Hence, minimizing $\tilde{\mathbb{P}}^{\text{tot}}$ will only rely on y_i and it is equal to $\min[(\sum_{i=1}^m \tilde{\mathcal{R}}_i^w = \sum_{i=1}^m y_i \times \frac{\sum_{k=1}^r |\tilde{\mathcal{R}}_i^k - \min(\tilde{\mathcal{R}}_i^k)| + \epsilon}{\sum_{k=1}^r \mathcal{U}_i^k}), \epsilon]$. The VMP problem can be simplified as

$$\min \left(\mathbb{R}^{\text{tot}} + \tilde{\mathbb{P}}^{\text{tot}}\right) \tag{14}$$

subject to

$$\sum_{i=1}^{m} x_{ij} = 1 \quad \forall j \in \boldsymbol{V} \tag{15}$$

$$\sum_{j=1}^{n} x_{ij} \times \tilde{\mathcal{R}}_i^k \leq C_i^k \quad \forall i \in \boldsymbol{P}, \forall k \in \boldsymbol{K} \tag{16}$$

under same control variable. This problem is like an already known *generalized assignment problem* (GAP) [39]. GAP problem tries to minimize the assignment costs of tasks to the agents under the consumed resource constraint by the tasks while we have limited agents. Similarly, in the VMP problem, the tasks are the VMs, and the agents are the PMs. Additionally, each VM has some limited resource demands that need to cover per assigned PM, by considering the maximum capability of each PM. Hence, in the updated problem, we aim to find the appropriate x_{ij}, which indicates the associated VMs to PMs. Since the GAP problem is NP-hard [39], and it includes the similarity-mapped VMP problem, we can conclude that the VMP is also NP-hard. ■

Algorithm 1: GRVMP Algorithm.

```
Input: $\boldsymbol{P}$, $\boldsymbol{V}$
Output: Placement of VMs on PMs
1:  Sort PMs in $\boldsymbol{P}$ in descending order of power
    efficiency through (6);
2:  while $\boldsymbol{V} \neq \emptyset$ do
3:      $\boldsymbol{S} = \{1, 2, \ldots, \min(|V|, d)\}$;
4:      Select $\boldsymbol{S}$ VMs from list $\boldsymbol{V}$, randomly, uniformly;
5:      for each $P_i \in \boldsymbol{P}$ do
6:          $MIN \leftarrow \infty$;
7:          for each $V_i \in \boldsymbol{S}$ do
8:              if ($R_j^k > C_j^k \; \forall k \in \boldsymbol{K}$) then
9:                  Calculate $\mathcal{R}_{ij}^w, i \in \boldsymbol{P}, j \in \boldsymbol{V}$ through
                    (17);
10:                 if $R_{ij}^w < MIN \; \forall i \in \boldsymbol{P}, j \in \boldsymbol{V}$ then
11:                     $MIN \leftarrow R_{ij}^w$;
12:                     $index \leftarrow j$;
13:                 end if
14:             end if
15:             if $MIN \neq \infty$ then
                  At least one VM can be
                  hosted on $P_i$
16:                 $C_i^k \leftarrow C_i^k - R_{index}^k, \forall i \in \boldsymbol{P}, k \in \boldsymbol{K}$;
17:                 $\boldsymbol{V} = \boldsymbol{V}/V_{index}$;
                  delete $V_{index}$ from $\boldsymbol{V}$
18:                 break
                    go to the next sampling $V_i$
19:             else
20:                 continue
                  go to the next PM $P_i$
21:             end if
22:         end for
23:     end for
24: end while
25: return Placement $\boldsymbol{V}$ on $\boldsymbol{P}$
```

IV. GRVMP Algorithm

In this section, we describe the proposed GRVMP in detail. We first describe our GRVMP (see Section IV-A). Then, we provide an illustrative example (see Section IV-B). Finally, we detail the time and space complexity of the GRVMP algorithm (see Section IV-C).

A. Proposed Greedy Algorithm

The proposed GRVMP is a greedy randomized approach for VMP that jointly minimizes the power consumption and resource wastage in a CDCN. To minimize the power consumption, GRVMP takes into account *two* main strategies. First, we order power-efficient PMs in descending manner. Formally speaking, we list the more power-efficient PMs as the highest priorities for the placement process. This strategy imposes the least increase in the energy consumption of a CDCN. Second, we need to decrease the number of active PMs as much as we can. This can be done by minimizing the resource wastage of each activated PM. We also reduce the PM resource wastage by maximizing the resource utilization of a PM.

To achieve the aforementioned aims, we first sort the list $\boldsymbol{P}$ based on their power efficiency in descending order. Then, we get the idea from the "*power of two choices*" model and for each step, the algorithm samples d VMs uniformly at random from the list $\boldsymbol{V}$. After that, the $\boldsymbol{P}$ is searched and the first PM that has enough resources to assign at least one of the d VMs will be chosen (this process is according to the FF strategy). If more than one VM can be hosted on the chosen PM, the VM with the minimum resource wastage is placed on that PM.

Algorithm 1 presents the pseudocode of the proposed GRVMP. In this algorithm, we first sort out the available PMs based on their power efficiency in decreasing order using (6) (see line #1). It is worth mentioning that we ignore the underutilized and overutilized PMs and we do not list them in $\boldsymbol{P}$. The algorithm executes the outer big loop (see lines 2–24) to assign all VMs to the most suitable PMs. To this end, we first select $\boldsymbol{S}$ set of VMs using the statement we list in line #3. In other words, we give a uniformly and randomly selected $\min(|V|, d)$ VMs as a sample

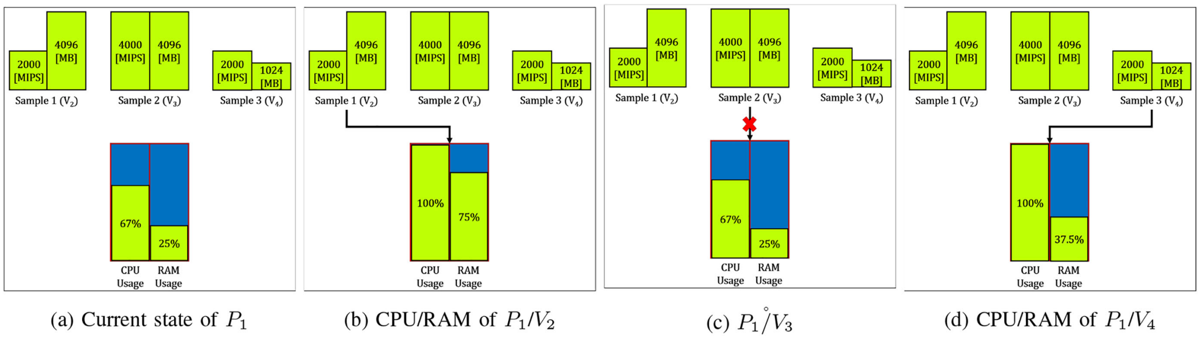

Fig. 2. Illustrative example (the total capacity of $P_1 = (6000$ [MIPS], 8192 [MB]). $A/B:=$ PM A after placing VM B; $A\mathring{/}B:=$ VM B cannot place on PM A.

to assign them to the available PMs (see line #4). Then, from the beginning of the list $\boldsymbol{P}$, we check all PMs until we find a PM that can host *at least* one of the sampled VMs, which is presented in $\boldsymbol{S}$ (see lines #5–#23). Among the sampled VMs, we select the one that has the minimum value of the resource wastage on P_i (see lines #5–#14). Then, we place the corresponding VM on the PM and then remove it (i.e., V_{index}) from the list $\boldsymbol{V}$ and update the resource of P_i (see lines #15–#18). However, if none of the sampled VMs can be hosted on P_i, the algorithm goes to the next PM (lines #19 and #20). In fact, GRVMP assigns the most suitable VM to each PM. The resource wastage of P_i when V_j is placed on it can be calculated as follows:

$$R^w_{ij} = \frac{\sum_{k=1}^{r} |\mathcal{R}^k_{ij} - \min(\mathcal{R}^k_{ij})| + \epsilon}{\sum_{k=1}^{r} U^k_{ij}}, \; i \in \boldsymbol{P}, j \in \boldsymbol{V} \quad (17)$$

where U^k_{ij} and $\mathcal{R}^k_{ij}$ are the normalized utilization and the normalized remaining resource of PM P_i across kth dimension after placing VM V_j on it, respectively.

B. Illustrative Example

We now illustrate how the proposed GRVMP algorithm works with an example. For an illustration purpose, we assume resources across two main dimensions, namely CPU and RAM, (i.e., $r = 2$). Also, we set the number of samples equal to three (i.e., $d = 3$). Let us consider a PM P_1 with the total capacity P_1 = (6000 [MIPS], 8192 [MB]) where a VM V_1 with the demand V_1 = (4000 [MIPS], 4096 [MB]) is already hosted on it. Let us suppose that three VMs are randomly sampled for placement such that V_2 = (2000 [MIPS], 4096 [MB]), V_3 = (4000 [MIPS], 2048 [MB]), and V_4 = (2000 [MIPS], 1024 [MB]). It is clear that V_3 cannot be hosted on P_1, whereas V_2 and V_4 have the potential of assigning to this PM. Hence, the proposed algorithm calculates the resource wastage for these two VMs, where $\mathcal{R}^w_{12}$=0.143 and $\mathcal{R}^w_{14} = 0.455$. Then, VM V_2 places on PM P_1, because its resource wastage value is the smallest. Fig. 2 presents the case presented in this example.

C. Time Complexity

We now analyze the time complexity of the proposed GRVMP algorithm as presented in Algorithm 1. In line #1, we sort the m PMs, which require $\mathcal{O}(m \log m)$, where m is the size of the set of PM, $\boldsymbol{P}$. The rest of the algorithm includes *three* nested loops. In the first loop, external *while* loop (lines 2–24), we require to place a set of n VMs on the PMs. In the second loop (lines 5–23), in the worst case, the algorithm needs to examine all the m PMs. In the final loop (lines 7–22), the algorithm requires to check at most d VMs and calculate $\mathcal{R}^w_{ij}$ for them. Hence, these three loops require $\mathcal{O}(n \times m \times d)$. Thus, the time complexity of GRVMP algorithm is equivalent to $\mathcal{O}(n \times m \times d + m \log m)$. It is worth mentioning that d is a constant value, where $d << n$.

TABLE III
CHARACTERISTICS OF PMs

CPU [MIPS]	Memory [MB]	MP [W]	Min Power [W]
[4000-10000]	[4096-16384]	[100-300]	[0.6-0.7] × MP

MP:= Max Power

V. Performance Evaluation

This section presents an exhaustive experiment and validates GRVMP against the cutting-edge methods. It consists of the simulation setup, simulation metrics, and describe our compared algorithms. Finally, it provides results for both synthetic and Amazon EC2 scenarios.

A. Simulation Setup

We consider the CDCN consists of different number of heterogeneous PMs having 2-D resource capacities: CPU and memory. The capacity of CPU and memory of each PM is randomly generated from the range [4000–10 000] [MIPS] and [4096–16 384] [MB], respectively. Also, the maximum power consumption of a PM is generated randomly in [100–300] [W]. For the minimum power consumption of the PM, it first takes a random value in [0.6–0.7] and then it is multiplied in the maximum power consumption of that PM. This implies that the power consumption of an activated but idle PM is equal to 60%–70% of its full utilization state [40]. We present the characteristics of PMs in Table III. The source code of our article is available in [41].

Focusing on VMs, we consider different numbers of heterogeneous VMs with CPU and memory requirements. Here, we

TABLE IV
CONFIGURATIONS FOR SYNTHETIC VMS

CPU [MIPS]	Memory [MB]
{500, 1000, 2000, 4000}	{512, 1024, 2048, 4096}

TABLE V
CONFIGURATIONS FOR AMAZON EC2 VMS

Type	CPU [MIPS]	Memory [MB]
Micro	500	613
Small	1000	1700
Medium	2000	3750
Large	2500	850

take into account *two* different scenarios: 1) synthetic VMs and 2) Amazon EC2 VMs. In the first scenario, each VM requires one CPU core with 500, 1000, 2000, or 4000 [MIPS], and 512, 1024, 2048, or 4096 [MB] of RAM (see Table IV). In the second scenario, we consider four different configurations of Amazon EC2 instances including *micro*, *small*, *medium*, and *large* types, where Table V demonstrates their configurations.

To conduct experimental evaluation, all simulations are coded in C++ environment. In order to have high confidence results, we executed 30 runs for each experiment and reported their average, maximum, and minimum value. We vary the number of VMs from 128 to 4096, where the number of PMs is fixed to 2000. Next, we fix the number of VMs to 1024 and vary the number of PMs from 500 to 4000. All experimental simulations were carried out on a PC with Intel Core Duo 2.00 GHz processor, 2.00 GB RAM, and Windows 7 operating system.

B. Simulation Metrics

We use *five* metrics, which are 1) the number of active PMs, 2) average CPU utilization, 3) average memory utilization, 4) total resource wastage, and 5) total power consumption to evaluate the performance of the GRVMP algorithm. These metrics are also used in [28]. Hence, we have the following.

1) *Number of active PMs:* We use this metric to show that how many PMs are required to host all VMs. This metric not only impacts on the energy consumption of a CDCN but it can impact on the operational and management costs.
2) *Average CPU and memory utilization:* These two metrics represent the efficiency of VMP algorithm in terms of resource utilization of activated PMs. High CPU and memory utilization of PMs in a CDCN, results in more efficient use of resources provided by cloud providers. Average CPU and memory utilization of a CDCN with z ($\forall z, z \leq m$) where z is the number of active PMs can be calculated by (18) and (19), respectively

$$C_z^{\text{avg}} = \frac{1}{z}\sum_{i=1}^{n} y_i \times \left(\sum_{j=1}^{n} \frac{x_{ij}\cdot R_j^{\text{cpu}}}{C_i^{\text{cpu}}}\right),\ z \in \boldsymbol{P} \quad (18)$$

$$M_z^{\text{avg}} = \frac{1}{z}\sum_{i=1}^{n} y_i \times \left(\sum_{j=1}^{n} \frac{x_{ij}\cdot R_j^{\text{mem}}}{C_i^{\text{mem}}}\right),\ z \in \boldsymbol{P}. \quad (19)$$

3) *Total resource wastage:* This metric not only focuses on the maximization of resource utilization but determines the load balancing among different resources of each activated PM. The value of this metric for a CDCN is obtained using (10).
4) *Total power consumption:* This metric is a key metric and mainly depends on the power efficiency of the activated PMs and their resource utilization along different dimensions. Equation (9) is defined to measure this important metric.

C. Compared Algorithms

We compare the performance of GRVMP algorithm against the following heuristic algorithms.

1) *First fit decreasing (FFD)* [18], [42]*:* This is a commonly used benchmark for evaluating the efficiency of VMP algorithms. First, we sort VMs in decreasing order based on their CPU demand and then for each VM, it searches the PM list from the beginning until to find a PM with adequate resource capacity.
2) *BFD* [42]*:* Similar to FFD, this strategy sorts VMs decreasingly according to their CPU requirement. Then, each VM is hosted on a PM that has enough resources and leave the least CPU resource.
3) *Random first fit (RFF):* This is a simple randomized strategy where it randomly selects VMs one by one from the VM list. Then it assigns each selected VM to the first available PM from PM list.
4) *Modified best fit decreasing (MBFD)* [3]*:* MBFD sorts VMs in decreasing order of their CPU demand and then selects a PM with the least increase in power consumption among all the available PMs to host the target VM.
5) *MaxMin* [14]*:* Similar to FFD, BFD, and MBFD, the MaxMin algorithm first decreasingly sorts VMs according to their CPU demands, then from the beginning of the VM list it attempts to host as many as possible VMs into the current activated PM and host as many as possible VMs from the end of VM list. When the current activated PM has not enough resources anymore, a new PM is activated.

D. Results

In this part, we provide the results for *two* aforementioned scenarios: synthetics (see Section V-D1) and Amazon EC2 real emulation test bed (see Section V-D2).

1) Synthetic Scenario: To specify our choice of d, we evaluate GRVMP's performance. To this end, we conducted an experiment in which the number of PMs and VMs are set to 2000 and 1024, respectively. Here, we report the results for three of the most important metrics and the average value of runs. As Table VI indicates, having only two choices significantly improves the performance of the proposed algorithm. Compared to the case where there is only one choice, the number of activated PMs is reduced by around 5%. Also, the total resources wastage and power consumption is improved by 25% and 7%, respectively. It is important to note that while a few choices dramatically improve performance, excessive amount of them

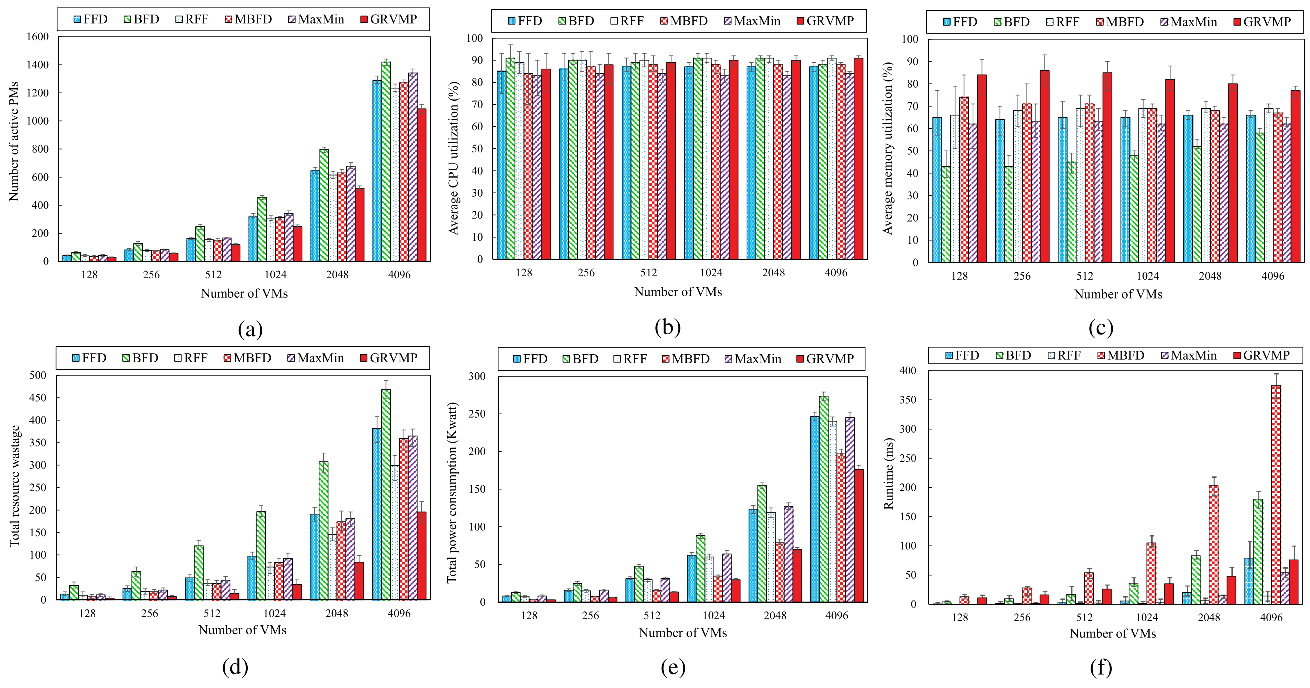

Fig. 3. Simulation results for 2000 PMs where (a) z:= number of active PMs, (b) C_z^{avg}:= average CPU utilization, (c) M_z^{avg}:= average memory utilization, (d) $\mathbb{R}^{\text{tot}}$:= total resource wastage, (e) $\mathbb{P}^{\text{tot}}$:= total resource wastage for various number of VMs, and (f) runtime.

TABLE VI
BEHAVIOR OF THE GRVMP FOR THE DIFFERENT VALUE OF PARAMETER d UNDER Synthetic Workload

d	**1**	**2**	**4**	**8**	**16**	**32**	**64**
z	260	**248**	251	254	254	257	258
$\mathbb{R}^{tot}$	46.15	**34.58**	37.08	38.09	37.62	37.71	37.22
$\mathbb{P}^{tot}$	31,833	**29,730**	30,099	30,399	30,287	30,603	30,703

z:= Number of active PMs, $\mathbb{R}^{\text{tot}}$:= Total resource wastage, and $\mathbb{P}^{\text{tot}}$:= Total power consumption.

does not have a significant impact. The reason is that the larger number of available choices makes it possible for the first PMs in the PM list to host VMs with their desirable configuration while not leaving VMs with desirable configuration for other PMs. This leads to more PMs being activated, and thus to more power consumption.

Experiment 1: PMs are fixed to 2000 where the number of VMs varies.

In this experiment, we fix the number of PMs to be 2000 (i.e., m=2000) and vary the number of VMs. Fig. 3 shows the superiority of the proposed GRVMP over existing algorithms almost in all aspects. Specially, Fig. 3(a) presents the number of active servers by varying the number of running VMs. It indicates that our GRVMP outperforms other algorithms in terms of the number of PMs needed to host VMs. In this figure, we can understand that MBFD and RFF, and then FFD and MaxMin use fewer number of PMs, respectively. BFD shows the worst performance among the others. This is because of its strategy to select PMs for assigning VMs, where it selects PMs with the lowest CPU resource capacity, which leads to the need for a large number of PMs. For this test, the percentage improvement of our GRVMP over the second-best results is about 12%–23%. Fig. 3(b) and (c) shows average CPU and memory utilization among the tested algorithms, respectively. From these two figures, we can conclude four points. First and the most important one is that our GRVMP is the *only* algorithm that provides high resource utilization in a load-balancing manner for *both CPU and memory resources*. Second, although BFD offers high CPU utilization, its memory utilization is extremely low (worst of all the compared algorithms). Third, the average CPU utilization in all algorithms is higher than the average memory utilization. This is due to the fact that the considered PMs provide higher memory than CPU. Fourth, both of our GRVMP and MBFD have the same behavior. Hence, when the number of VMs increases, the average CPU utilization slightly increases, whereas the average memory utilization value somewhat decreases. To justify this phenomenon, we should mention that both GRVMP and MBFD host VMs on the PMs with almost higher CPU resources. Thus by increasing the number of VMs, the possibility of hosting VMs with high CPU requirements on PMs with high CPU capacity increases. Focusing on Fig. 3(d), we illustrate the total resource wastage of the considered CDCN. The figure depicts that the total resource wastage of the proposed GRVMP is significantly less than the others. This significant superiority is mainly because of our strategy for placing randomly selected VMs on PMs [see (17)]. After the proposed algorithm, RFF works better. This is due to fact that FFD, BFD, MBFD, and MaxMin algorithms only take into account CPU resource utilization while random selection of VMs helps RFF to have less resource wastage. Here, BFD has the worst performance. Fig. 3(e) shows results for the total power consumption. MBFD is our main competitor in this test. GRVMP exhibits 11%–17%

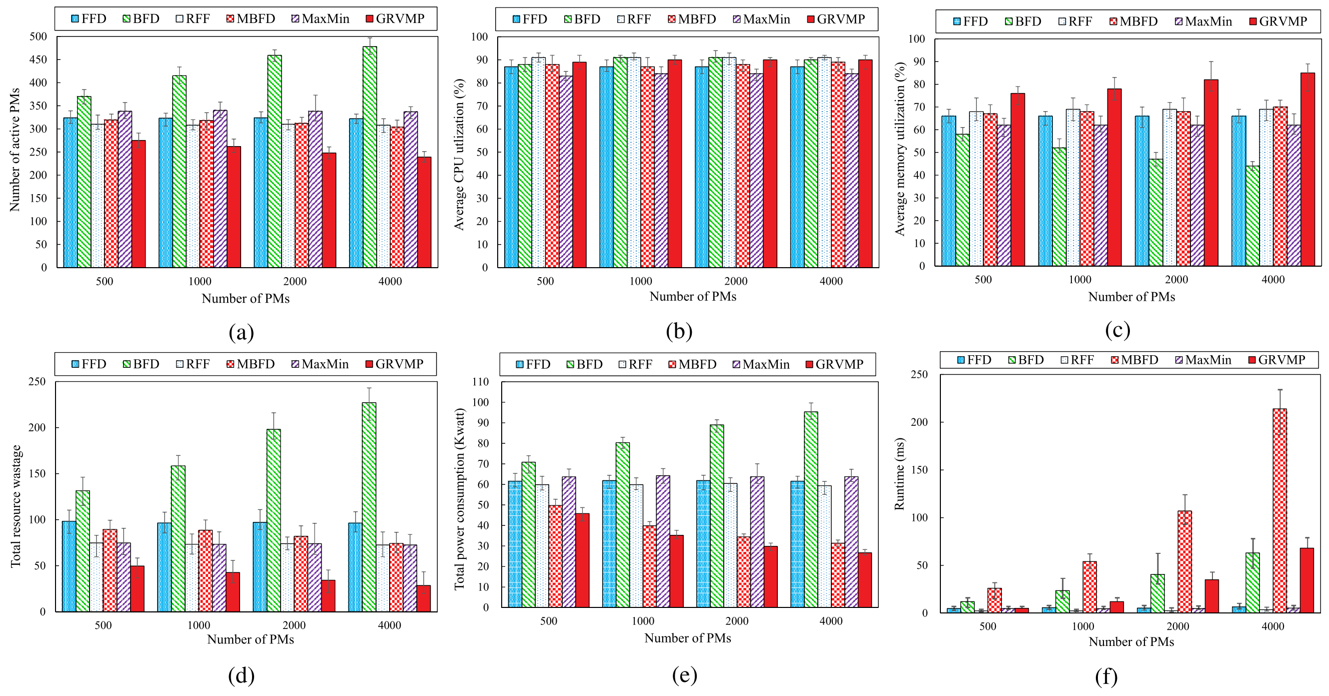

Fig. 4. Simulation results for 1024 VMs where (a) z:= number of active PMs, (b) C_z^{avg}:= average CPU utilization, (c) M_z^{avg}:= average memory utilization, (d) $\mathbb{R}^{\text{tot}}$:= total resource wastage, (e) $\mathbb{P}^{\text{tot}}$:= total resource wastage for various number of PMs, and (f) runtime.

improvement in total power consumption compared to MBFD. The power consumption of FFD, RFF, and MaxMin is almost the same, whereas BFD consumes the most power. Finally, Fig. 3(f) illustrates the runtime results of the compared algorithms. As we can see, all algorithms provide low execution time. However, the execution time of MBFD and then BFD is higher than the others for a large number of VMs. RFF, MaxMin, FFD, and the proposed GRVMP perform very well even when a CDCN receives a lot of VM requests.

Experiment 2: VMs are fixed to 1024, whereas the number of PMs varies.

In this scenario, we have an experiment that fixes the number of VMs to 1024 (i.e., n=1024) and varies the number of PMs and tests our method compared to the state-of-the-art methods. Fig. 4 shows the superiority of the proposed GRVMP over existing algorithms almost in all aspects.

Fig. 4(a)–(e) illustrates the simulation results of the *second* experiment. As we can see from these figures, our GRVMP gives the best results in all cases. From Fig. 4(a), it is clear that the proposed algorithm uses far fewer PMs than the others. As the number of PMs increase, the percentage of improvement also increases. For instance, compared to RFF, our GRVMP reduces the number of active PMs from 11% to 22%, and compared to MBFD the amount of improvement is from 14% to 21%. It is worth mentioning that since GRVMP sorts PMs by their power efficiency, with more PMs available it has a higher chance to host VMs on PMs with more computational power. However, this is opposite for BFD.

Focusing on Fig. 4(b) and (c), GRVMP performs very well on average CPU and memory utilization. Although almost all algorithms provide high average CPU utilization, the proposed

TABLE VII
BEHAVIOR OF THE GRVMP FOR THE DIFFERENT VALUE OF PARAMETER d UNDER `Amazon EC2 Workload`

d	1	2	4	8	16	32	64
z	222	207	**201**	206	210	212	214
$\mathbb{R}^{tot}$	40.09	21.19	**13.44**	14.84	16.39	17.85	17.73
$\mathbb{P}^{tot}$	25,692	23,817	**23,407**	23,724	24,206	24,345	24,556

z:= Number of active PMs, $\mathbb{R}^{\text{tot}}$:= Total resource wastage, and $\mathbb{P}^{\text{tot}}$:= Total power consumption.

algorithm also gives high memory utilization. By increasing the number of PMs, our algorithm increases the average memory utilization, whereas BFD decreases it. That is because the increasing number of PMs makes it more likely to have PMs with low CPU and high memory capacities; then, BFD selects such PMs that result in high memory wastage. However, the strategy used in our algorithm benefits from these diversities. Focusing on Fig. 4(d), we can obviously see that our GRVMP dramatically reduces resource wastage. As the number of PMs increase the reduction degree decreases. This behavior is also true for MBFD, whereas it is opposite for BFD. This is due to the randomness of the dataset. Increasing the number of PMs makes it more possible for our GRVMP and MBFD to select more suitable PMs for hosting VMs on them, whereas BFD selects PMs with low CPU and high memory capacities, which lead to more unbalanced resource utilization. Fig. 4(e) shows the results for the total power consumption of a CDCN. As we can observe, our GRVMP performs the best, MBFD gives the second, FFD, RFF, and MaxMin have almost the same results and give the third-best results. BFD has the worst rank. GRVMP reduces the power consumption from 8% to 15% compared with

TABLE VIII
PERCENTAGE (%) IMPROVEMENT OF OUR GRVMP Algorithm Compared With Literature FOR EXPERIMENT #3 REPORTED IN SECTION V-D2 (z:= NUMBER OF ACTIVATED PMs, $\mathbb{R}^{\text{tot}}$:= RESOURCE WASTAGE, AND $\mathbb{P}^{\text{tot}}$:= POWER CONSUMPTION)

	FFD			BFD			RFF			MBFD			MaxMin		
#VMs	z	$\mathbb{R}^{tot}$	$\mathbb{P}^{tot}$	z	$\mathbb{R}^{tot}$	$\mathbb{P}^{tot}$	z	$\mathbb{R}^{tot}$	$\mathbb{P}^{tot}$	z	$\mathbb{R}^{tot}$	$\mathbb{P}^{tot}$	z	$\mathbb{R}^{tot}$	$\mathbb{P}^{tot}$
128	26.47%	74.22%	61.06%	51.92%	86.58%	75.44%	24.24%	64.78%	60.99%	21.88%	67.47%	16.46%	30.56%	77.01%	63.32%
256	27.94%	76.90%	60.56%	51.49%	87.34%	74.01%	25.76%	67.39%	59.47%	23.44%	71.96%	18.03%	31.94%	79.35%	62.22%
512	28.26%	79.61%	58.74%	51.71%	88.90%	72.64%	22.66%	69.86%	55.93%	24.43%	75.69%	17.74%	30.28%	80.52%	59.07%
1024	27.17%	82.82%	54.89%	46.68%	89.22%	67.85%	22.69%	75.63%	52.81%	25.00%	81.11%	17.90%	30.45%	84.75%	56.39%
2048	23.58%	78.53%	47.87%	39.60%	85.12%	59.89%	19.62%	69.41%	46.03%	22.59%	77.37%	16.53%	27.56%	81.14%	49.88%
4096	19.73%	71.88%	36.12%	30.68%	79.75%	45.38%	15.07%	60.12%	33.28%	19.36%	71.17%	16.09%	23.83%	75.08%	38.41%

TABLE IX
PERCENTAGE (%) IMPROVEMENT OF OUR GRVMP Algorithm Compared With Literature FOR EXPERIMENT #4 REPORTED IN SECTION V-D2 (z:= NUMBER OF ACTIVATED PMs, $\mathbb{R}^{\text{tot}}$:= RESOURCE WASTAGE, AND $\mathbb{P}^{\text{tot}}$:= POWER CONSUMPTION)

	FFD			BFD			RFF			MBFD			MaxMin		
#PMs	z	$\mathbb{R}^{tot}$	$\mathbb{P}^{tot}$	z	$\mathbb{R}^{tot}$	$\mathbb{P}^{tot}$	z	$\mathbb{R}^{tot}$	$\mathbb{P}^{tot}$	z	$\mathbb{R}^{tot}$	$\mathbb{P}^{tot}$	z	$\mathbb{R}^{tot}$	$\mathbb{P}^{tot}$
500	20.36%	71.01%	36.36%	31.13%	79.35%	45.58%	16.41%	59.15%	33.56%	20.07%	70.41%	15.16%	24.48%	74.20%	38.30%
1000	23.44%	77.93%	47.56%	39.77%	84.82%	59.90%	20.23%	69.12%	45.96%	22.30%	76.89%	15.77%	27.43%	80.61%	49.66%
2000	26.28%	79.93%	54.24%	46.28%	87.35%	67.84%	21.71%	70.62%	52.66%	24.0%	77.50%	16.99%	29.86%	82.26%	56.16%
4000	27.74%	81.46%	58.51%	51.71%	90.20%	72.77%	24.14%	73.84%	57.05%	24.14%	78.03%	18.02%	31.49%	83.73%	59.87%

MBFD. Similar to Fig. 4(a) and (d), by increasing the number of available PMs, our GRVMP considerably consumes less power. Again, this is because of our strategy for selecting PMs where we sort PMs based on their power efficiency. Fig. 4(f) shows that when the number of available PMs is low, our GRVMP provides very low running time, comparable to RFF, MaxMin, and FFD. However, as the number of available servers increases the runtime of GRVMP somewhat increases. This is because of sorting PMs based on their power efficiency.

2) Amazon EC2 Scenario: In the second scenario, we carry out an experiment to see how the parameter d impacts the results, where we set the number of PMs and VMs to 2000 and 1024, respectively. Table VII shows the results. Here, we can observe that $d = 4$ gives the best results. So, we set $d = 4$ throughout all tests done for Amazon EC2 scenarios. It is worth mentioning that for this scenario, the behavior of the algorithms in terms of the average CPU and memory utilization almost is similar to the synthetic scenario. Hence, they are not listed.

Experiment 3: PMs are fixed to 2000, whereas the number of VMs varies.

In this scenario, we run the experiment for 2000 PMs (i.e., m=2000) and vary the number of VMs and compare our method to the state-of-the-art methods.

Table VIII demonstrates the improvement of GRVMP over the others with regard to the number of activated PMs, total resource wastage, and total power consumption. Focusing on the number of activated PMs, the improvement of GRVMP is over 15% for all the compared algorithms and achieves up to 51%. Focusing on the resource wastage, GRVMP enhancement rate is drastically where it is between 60% and 89%. The improvement in power consumption is above 33% compared to FFD, BFD, RFF, and MaxMin where the best improvement reaches 75%. GRVMP shows 16%–18% improvements in total power saving when compared to MBFD. We should mention an *important point* here. Although the improvement of our GRVMP is significant, however, it is clear from Table VIII that as the number of VMs increases, the percentage of improvement decreases. We have dealt with this issue in depth and found that when the number of VMs grows, GRVMP has to select some less power-efficient PMs. It means PMs with less CPU capacity and higher power consumption. Let us remind that our GRVMP sorts PMs based on their power efficiency. However, when the number of VMs is low, the proposed algorithm has the opportunity to host them on the more power-efficient PMs. It is worth mentioning that the runtime behavior of the algorithms is similar to Fig. 3(f); so we do not report it.

Experiment 4: VMs are fixed to 1024, whereas the number of PMs varies.

In this scenario, we run the experiment for 1024 VMs (i.e., n=1024) and vary the number of PMs and compare our method to the state-of-the-art methods. The results are reported in Table IX. In this table, the percentage of improvement in the number of used PMs varies from 20% to 27% over FFD, from 31% to 51% over BFD, from 16% to 24% over RFF, from 20% to 24% over MBFD, and from 24% to 31% over MaxMin. In terms of resource wastage, the proposed algorithm shows great improvement compared to the other algorithms. This amount of improvement is to be expected because our GRVMP places VMs on PMs based on these important metrics, whereas others do not consider it. RFF has the second-best results on resource wastage. Considering power consumption, GRVMP performs very well and provides a significant improvement over all other algorithms. The main reason behind this improvement is that none of FFD, BFD, RFF, and MaxMin take into account power consumption during VM placement. Although MBFD considers this key metric, it ignores the efficient use of resources of activated PMs. Unlike experiment #3 reported in Section V-D2, in this experiment we increase the number of PMs, the improvement percentage of GRVMO increases for all metrics. To justify this behavior, the higher number of available PMs gives our algorithm the opportunity of placing VMs on the more power-efficient and powerful PMs. Here again, algorithms give almost the similar runtime to those presented in Fig. 4(f).

VI. Discussion and Limitations

Despite the effectiveness of our GRVMP, several aspects are remaining that this section addresses them. First, similar to many of the other works [3], [13], [28], the presented work does not consider the network topology of a CDCN and flow among VMs on the application level. Although some recent research works have focused on this aspect [43]–[45], how to minimize power consumption, resource wastage, and network bandwidth consumption has remained as a key challenge in today's CDCNs. To address this issue, our problem formulation must be extended to build a suitable model for network bandwidth consumption. The model should guarantee the proximity of PMs hosted VMs of an application. Second, our primary goal in this work is to focus on the VM placement problem. Therefore, we need to add some other phases to our algorithm to apply to the VM migration problem. In particular, we believe that our randomized approach can apply to the VM selection phase, with a little modification. Finally, due to the low time complexity and the high performance of the proposed GRVMP, we can use it in a dynamic service placement problem in the domain of fog computing [46], [47].

VII. Conclusion

In this article, we proposed a greedy randomized algorithm for VM placement in large-scale CDCNs. We evaluated the performance of the proposed algorithm with the state-of-the-art algorithms through extensive simulation experiments. The results demonstrate that our algorithm significantly reduces the number of active PMs, the total resource wastage, and the total power consumption of CDCNs with different configurations. The results of the experiment show that the improvement of our algorithm is above 15% for all baseline algorithms and achieves up to 51%, in terms of the number of activated PMs. Regarding the resource wastage, the percentage of improvement is significant, where it is between 59% and 90%. In comparison with the second-best results, our algorithm exhibits 15%–18% gain in total power efficiency. For future work, We intend to apply ensemble learning or federate machine learning solutions to tackle the VMP problem in the cloud system.

References

[1] Z. Zhou *et al.*, "Minimizing SLA violation and power consumption in cloud data centers using adaptive energy-aware algorithms," *Future Gener. Comput. Syst.*, vol. 86, pp. 836–850, 2018.

[2] M. Masdari, S. S. Nabavi, and V. Ahmadi, "An overview of virtual machine placement schemes in cloud computing," *J. Netw. Comput. Appl.*, vol. 66, pp. 106–127, 2016.

[3] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing," *Future Gener. Comput. Syst.*, vol. 28, no. 5, pp. 755–768, 2012.

[4] W. Lin, W. Wu, and L. He, "An on-line virtual machine consolidation strategy for dual improvement in performance and energy conservation of server clusters in cloud data centers," *IEEE Trans. Serv. Comput.*, vol. 8, no. 4, pp. 1162–1175, Oct.-Dec. 2020.

[5] M. Shojafar, C. Canali, R. Lancellotti, and J. Abawajy, "Adaptive computing-plus-communication optimization framework for multimedia processing in cloud systems," *IEEE Trans. Cloud Comput.*, to be published, doi: 10.1109/TCC.2016.2617367.

[6] C. Wei, Z.-H. Hu, and Y.-G. Wang, "Exact algorithms for energy-efficient virtual machine placement in data centers," *Future Gener. Comput. Syst.*, vol. 106, pp. 77–91, 2020.

[7] H. Talebian *et al.*, "Optimizing virtual machine placement in IaaS data centers: Taxonomy, review and open issues," *Cluster Comput.*, 2019.

[8] F. L. Pires and B. Barán, "A virtual machine placement taxonomy," in *Proc. 15th IEEE/ACM Int. Symp. Cluster, Cloud Grid Comput.*, 2015, pp. 159–168.

[9] W. Attaoui and E. Sabir, "Multi-criteria virtual machine placement in cloud computing environments: A literature review," 2018, *arXiv:1802.05113*.

[10] Z. A. Mann and M. Szabó, "Which is the best algorithm for virtual machine placement optimization?" *Concurrency Comput., Pract. Experience*, vol. 29, no. 10, 2017, Art. no. e4083.

[11] A. Anand, J. Lakshmi, and S. Nandy, "Virtual machine placement optimization supporting performance SLAs," in *Proc. IEEE 5th Int. Conf. Cloud Comput. Technol. Sci.*, 2013, vol. 1, pp. 298–305.

[12] Q. Zheng *et al.*, "Virtual machine consolidated placement based on multi-objective biogeography-based optimization," *Future Gener. Comput. Syst.*, vol. 54, pp. 95–122, 2016.

[13] F. Alharbi, Y.-C. Tian, M. Tang, W.-Z. Zhang, C. Peng, and M. Fei, "An ant colony system for energy-efficient dynamic virtual machine placement in data centers," *Expert Syst. Appl.*, vol. 120, pp. 228–238, 2019.

[14] D. Zhu, "Max–min bin packing algorithm and its application in nano-particles filling," *Chaos, Solitons Fractals*, vol. 89, pp. 83–90, 2016.

[15] S. Jangiti and V. S. Shankar Sriram, "Scalable and direct vector bin-packing heuristic based on residual resource ratios for virtual machine placement in cloud data centers," *Comput. Elect. Eng.*, vol. 68, pp. 44–61, 2018.

[16] S. Azizi, M. Zandsalimi, and D. Li, "An energy-efficient algorithm for virtual machine placement optimization in cloud data centers," *Cluster Comput.*, pp. 1–14, 2020.

[17] M. Mitzenmacher, "The power of two choices in randomized load balancing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 12, no. 10, pp. 1094–1104, Oct. 2001.

[18] G. Keller, M. Tighe, H. Lutfiyya, and M. Bauer, "An analysis of first fit heuristics for the virtual machine relocation problem," in *Proc. 8th Int. Conf. Netw. Serv. Manage./Workshop Syst. Virtualiztion Manage.*, 2012, pp. 406–413.

[19] M. Masdari and M. Zangakani, "Green cloud computing using proactive virtual machine placement: Challenges and issues," *J. Grid Comput.*, pp. 1–33, 2019.

[20] Y. Wu, M. Tang, and W. Fraser, "A simulated annealing algorithm for energy efficient virtual machine placement," in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, 2012, pp. 1245–1250.

[21] S. Wang, Z. Liu, Z. Zheng, Q. Sun, and F. Yang, "Particle swarm optimization for energy-aware virtual machine placement optimization in virtualized data centers," in *Proc. Int. Conf. Parallel Distrib. Syst.*, 2013, pp. 102–109.

[22] A. Al-Moalmi, J. Luo, A. Salah, and K. Li, "Optimal virtual machine placement based on grey wolf optimization," *Electronics*, vol. 8, no. 3, pp. 1–22, 2019.

[23] Y. Gao, H. Guan, Z. Qi, Y. Hou, and L. Liu, "A multi-objective ant colony system algorithm for virtual machine placement in cloud computing," *J. Comput. Syst. Sci.*, vol. 79, no. 8, pp. 1230–1242, 2013.

[24] X.-F. Liu, Z.-H. Zhan, J. D. Deng, Y. Li, T. Gu, and J. Zhang, "An energy efficient ant colony system for virtual machine placement in cloud computing," *IEEE Trans. Evol. Comput.*, vol. 22, no. 1, pp. 113–128, Feb. 2018.

[25] E. Parvizi and M. H. Rezvani, "Utilization-aware energy-efficient virtual machine placement in cloud networks using NSGA-III meta-heuristic approach," *Cluster Comput.*, pp. 1–23, 2020.

[26] A. Abohamama and E. Hamouda, "A hybrid energy–aware virtual machine placement algorithm for cloud environments," *Expert Syst. Appl.*, vol. 150, 2020, Art. no. 113306.

[27] X. Li, Z. Qian, S. Lu, and J. Wu, "Energy efficient virtual machine placement algorithm with balanced and improved resource utilization in a data center," *Math. Comput. Model.*, vol. 58, no. 5/6, pp. 1222–1235, 2013.

[28] M. K. Gupta and T. Amgoth, "Resource-aware virtual machine placement algorithm for IaaS cloud," *J. Supercomput.*, vol. 74, no. 1, pp. 122–140, 2018.

[29] W. Yao, Y. Shen, and D. Wang, "A weighted pagerank-based algorithm for virtual machine placement in cloud computing," *IEEE Access*, vol. 7, pp. 176369–176381, 2019.

[30] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 4, pp. 63–74, 2008.

[31] C. Chekuri and S. Khanna, "On multi-dimensional packing problems," in *Proc. 10th Annu. ACM-SIAM Symp. Discr. Algorithms*, 1999, pp. 185–194.

[32] H. Zhao, J. Wang, F. Liu, Q. Wang, W. Zhang, and Q. Zheng, "Power-aware and performance-guaranteed virtual machine placement in the cloud," *IEEE Trans. Parallel Distrib. Syst.*, vol. 29, no. 6, pp. 1385–1400, Jun. 2018.

[33] A.-C. Orgerie, L. Lefevre, and J.-P. Gelas, "Demystifying energy consumption in grids and clouds," in *Proc. Int. Conf. Green Comput.*, 2010, pp. 335–342.

[34] D. Kliazovich, P. Bouvry, and S. U. Khan, "Greencloud: A packet-level simulator of energy-aware cloud computing data centers," *J. Supercomput.*, vol. 62, no. 3, pp. 1263–1283, 2012.

[35] B. Krishnan, H. Amur, A. Gavrilovska, and K. Schwan, "VM power metering: Feasibility and challenges," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 38, no. 3, pp. 56–60, 2011.

[36] Y. C. Lee and A. Y. Zomaya, "Energy efficient utilization of resources in cloud computing systems," *J. Supercomput.*, vol. 60, no. 2, pp. 268–280, 2012.

[37] K.-Y. Chen, Y. Xu, K. Xi, and H. J. Chao, "Intelligent virtual machine placement for cost efficiency in geo-distributed cloud systems," in *Proc. IEEE Int. Conf. Commun.*, 2013, pp. 3498–3503.

[38] M. F. Bari *et al.*, "Data center network virtualization: A survey," *IEEE Commun. Surv. Tut.*, vol. 15, no. 2, pp. 909–928, Apr.–Jun. 2013.

[39] R. M. Nauss, "Solving the generalized assignment problem: An optimizing and heuristic approach," *INFORMS J. Comput.*, vol. 15, no. 3, pp. 249–266, 2003.

[40] A. M. Sampaio, J. G. Barbosa, and R. Prodan, "PIASA: A power and interference aware resource management strategy for heterogeneous workloads in cloud data centers," *Simul. Model. Pract. Theory*, vol. 57, pp. 142–160, 2015.

[41] S. Azizi, M. Shojafar, J. Abawajy, and R. Buyya, "GRVMP source code," 2020. [Online]. Available: https://github.com/mshojafar/sourcecodes/blob/master/IEEEISJVMP2020Sadoon_Sourcecode.zip

[42] M. A. Khan, A. Paplinski, A. M. Khan, M. Murshed, and R. Buyya, "Dynamic virtual machine consolidation algorithms for energy-efficient cloud resource management: A review," in *Sustainable Cloud and Energy Services*. Berlin, Germany: Springer-Verlag, 2018, pp. 135–165.

[43] S.-H. Wang, P. P.-W. Huang, C. H.-P. Wen, and L.-C. Wang, "EQVMP: Energy-efficient and QoS-aware virtual machine placement for software defined datacenter networks," in *Proc. Int. Conf. Inf. Netw.*, 2014, pp. 220–225.

[44] J. Son and R. Buyya, "Priority-aware VM allocation and network bandwidth provisioning in software-defined networking (SDN)-enabled clouds," *IEEE Trans. Sustain. Comput.*, vol. 4, no. 1, pp. 17–28, Jan.–Mar. 2018.

[45] G. Luo, Z. Qian, M. Dong, K. Ota, and S. Lu, "Improving performance by network-aware virtual machine clustering and consolidation," *J. Supercomput.*, vol. 74, no. 11, pp. 5846–5864, 2018.

[46] A. Yousefpour *et al.*, "FOGPLAN: A lightweight QoS-aware dynamic fog service provisioning framework," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 5080–5096, Jun. 2019.

[47] S. Azizi, H. Omer, and M. Shojafar, "Priority, network and energy-aware placement of IoT-based application services in fog-cloud environments," *IET Commun.*, pp. 1–13, 2020.

Sadoon Azizi (Member, IEEE) received the M.Sc. degree in computer science, with focus on high performance computing, and the Ph.D. degree in computer science, with focus on cloud data centers, from Amirkabir University of Technology, Tehran, Iran, in 2012 and 2016, respectively.

He is currently an Assistant Professor with the Department of Computer Engineering, University of Kurdistan, Sanandaj, Iran. He is also the Leader of the Internet of Things Research Laboratory and Manager with the High Performance Computing Center, University of Kurdistan. His current research interests include cloud computing, fog computing, Internet of Things, and heuristic algorithms.

Mohammad Shojafar (Senior Member, IEEE) received the Ph.D. degree in information and communications technology from the Sapienza University of Rome, Rome, Italy, in 2016.

He is currently a Senior Lecturer (Associate Professor) with the Network Security and an Intel Innovator with the 5G Innovation Centre (5GIC), University of Surrey, Guildford, U.K. Before joining 5GIC, he was a Senior Researcher and a Marie Curie Fellow with the University of Padua, Padua, Italy. He was a PI of PRISENODE project, a 275 000 Euro Horizon 2020 Marie Curie project in the areas of fog/cloud security collaborating with the University of Padua.

Dr. Shojafar is an Associate Editor for the IEEE TRANSACTIONS ON CONSUMER ELECTRONICS and *IET Communications*.

Jemal Abawajy (Senior Member, IEEE) received the B.Sc. degree from St. F. X University, Canada, the M.Sc. degree from Dalhousie University, Canada, and the Ph.D. degree from the Ottawa-Carleton Institute of Technology, Canada, all in computer science.

He is a Full Professor with the School of Information Technology, Deakin University, Burwood, VIC, Australia. He is currently the Director of the Parallel and Distributing Computing Laboratory. He is the author/co-author of five books, more than 250 papers in conferences, book chapters, and journals such as the IEEE TRANSACTIONS ON COMPUTERS and IEEE TRANSACTIONS ON FUZZY SYSTEMS.

Prof. Abawajy is a Senior Member of IEEE Computer Society, IEEE Technical Committee on Scalable Computing, IEEE Technical Committee on Dependable Computing and Fault Tolerance, and IEEE Communication Society.

Rajkumar Buyya received the B.E. degree in computer science from Mysore University, Mysore, India, in 1992, the M.E. degree in engineering from Bangalore University, Bangaluru, India, in 1995, and the Ph.D. degree in computer science and software engineering from Monash University, Melbourne, Australia, in 2002.

He is a Redmond Barry Distinguished Professor and Director of the Cloud Computing and Distributed Systems Laboratory, University of Melbourne, Melbourne, VIC, Australia. He has authored more than 750 publications and seven text books including *Mastering Cloud Computing* published by McGraw Hill, China Machine Press, and Morgan Kaufmann for Indian, Chinese, and international markets, respectively.

Dr. Buyya is one of the highly cited authors in computer science and software engineering worldwide. He was the Founding Editor-in-Chief for the IEEE TRANSACTIONS ON CLOUD COMPUTING. He is currently the Editor-in-Chief for *Software: Practice and Experience*.