# A Priority, Power and Traffic-aware Virtual Machine Placement of IoT Applications in Cloud Data Centers

Shvan Omer[a], Sadoon Azizi[a], Mohammad Shojafar[b,*], Rahim Tafazolli[b]

[a]*Department of Computer Engineering and Information Technology, University of Kurdistan, Sanandaj, Iran*
[b]*Institute for Communication Systems, 6G Innovation Centre, University of Surrey, Guildford, GU27XH, United Kingdom*

**Abstract**

Recent telecommunication paradigms, such as big data, Internet of Things (IoT), ubiquitous edge computing (UEC), and machine learning, are encountering with a tremendous number of complex applications that require different priorities and resource demands. These applications usually consist of a set of virtual machines (VMs) with some predefined traffic load between them. The efficiency of a cloud data center (CDC) as prominent component in UEC significantly depends on the efficiency of its VM placement algorithm applied. However, VM placement is an NP-hard problem and thus there exist practically no optimal solution for this problem. In this paper, motivated by this, we propose a priority, power and traffic-aware approach for efficiently solving the VM placement problem in a CDC. Our approach aims to jointly minimize power consumption, network consumption and resource wastage in a multi-dimensional and heterogeneous CDC. To evaluate the performance of the proposed method, we compared it to the state-of-the-art on a fat-tree topology under various experiments. Results demonstrate that the proposed method is capable of reducing the total network consumption up to 29%, the consumption of power up to 18%, and the wastage of resources up to 68%, compared to the second-best results.

*Keywords:* Cloud Computing, Internet of Thing (IoT), Cloud Data Center (CDC), Virtual Machine Placement (VMP), Priority-aware and Traffic-aware, Power Consumption.

## 1. Introduction

Cloud computing has become a promising platform for various applications such as big data analysis [1], Internet of Things (IoT) [2], machine learning [3], etc. Different applications require different resources in terms of computing power, storage space, and network bandwidth. Moreover, the quality-of-service (QoS) of applications also are different [4]. For example, applications like vehicular networks [5], deadline constraint scientific workflows [6], and face detection and tracking [7] require high QoS, such as minimal response time. Consequently, cloud providers (CPs) should provision resources in a way that these various requirements are satisfied.

CPs provision their resources in the form of virtual machine (VM) instances and allocate them to cloud users on a pay-as-you-go basis. Each cloud user requests a set of VMs to run its application on a cloud data center (CDCs), where VMs usually communicate to each other to process the application. Although virtualization technology allows cloud providers to run multiple VMs on a physical machine (PM), how to place the VMs of different applications on a pool of PMs is a challenging task. Efficient virtual machine placement (VMP) allows CPs to maximize their profit through minimizing the energy consumption of CDC, reducing the resource wastage of PMs, and maximizing the QoS offered to cloud users [8].

VMP is an NP-hard combinatorial optimization problem [9] for which there exist no optimal solution in practice. Therefore, in the literature numerous heuristics and metaheuristics have been applied for this problem [10, 11, 12].

*Corresponding author
*Email addresses:* shvan.omer@eng.uok.ac.ir (Shvan Omer), s.azizi@uok.ac.ir (Sadoon Azizi), m.shojafar@surrey.ac.uk (Mohammad Shojafar), r.tafazolli@surrey.ac.uk (Rahim Tafazolli)

The VMP problem has many perspectives such as the power consumption, resource wastage, traffic among VMs, and priority of applications. A vast majority of the proposed algorithms have focused only on reducing energy consumption [13, 14, 15, 16, 17]. Some of them have formulated the VMP as a bi-objective problem, considering the minimization of energy consumption and resource wastage [18, 19, 20]. A few of them have taken into account the network connection between VMs of a requested application [21, 22, 23]. The priority of applications has been considered only by the authors of [4]. To the best of our knowledge, this is the first effort that addresses all the above perspectives altogether.

Several questions arise addressing the priority and traffic-aware VMP in a multi-dimensional, heterogeneous CDC. How we can provide high QoS for IoT applications with higher priority? How the power consumption, resource wastage and network consumption of a CDC can be optimized. Can we assure that the proposed approach could provide low time complexity?

### *1.1. The goal of the paper and contributions*

In this work, we propose a novel priority-aware VMP algorithm that jointly optimizes the consumption of power, wastage of resource and consumption of network resources in a CDC with multidimensional and heterogeneous resources. Based on the priority level of a requested IoT application (critical or normal), we propose two efficient heuristic algorithms. For critical applications, a Joint Power and Traffic optimization algorithm, called *JointPT*, is presented. JointPT is an energy-efficient and QoS-aware approach which has two main goals including minimizing the power consumption by exploiting the power-efficient PMs and reducing the network consumption by allocating mutual VMs with the higher traffic demand into PMs with the closest proximity. For normal applications, we present a Joint Power and Resource optimization algorithm, called *JointPR*. The key objectives of JointPR are minimizing the total power consumption and resource wastage of a CDC by hosting each VM of a normal application on the power-efficient PM with sufficient resources and the least resource wastage.

The major contribution of this work are as follows:

- The VMP problem is formulated as a mixed integer linear programming (MILP) model with three objectives, i.e., the consumption of power, wastage of resources and consumption of network resources of a CDC.
- To efficiently solve the VMP problem in a CDC with multidimensional and heterogeneous resources, a priority-aware scheme is proposed.
- Various experiments are conducted to evaluate the performance of the proposed algorithm. In comparison with the state-of-the-art, results reveal that the proposed algorithm is a promising solution for the energy, resource, and network-aware VMP in CDC of UEC.

### *1.2. Roadmap*

Related works are discussed in Section 2. The proposed system architecture and problem formulation are described in Section 3 and , Section 4, respectively. In Section 5, the proposed algorithm is presented. Experimental evaluation of the proposed algorithm and analysis of the results are given in Section 6. The limitation of our work is discussed in Section 7. Finally, Section 8 concludes the paper and highlights some future research directions.

## 2. Related Work

In this section, we explain different research works that associated to the virtual machine placement problem. The background of the problem and the analysis of the state-of-the-art strategies can be found in [8, 10, 11]. Generally speaking, VM placement algorithms can be categorized into network-agnostic and network-ware. Network-agnostic algorithms do not consider the communication traffic between VMs and network connection among PMs, whereas network-aware ones take into account them. In the following, we briefly discuss some of the most relevant ones, with focus on their main contributions.

### 2.1. Network-agnostic VMP

Modified best fit decreasing (MBFD) [13] is the most well-known heuristic algorithm that has been proposed for VM placement. MBFD first sorts VMs based on their resource requirement and then place each of them on a PM with the least increase in the energy consumption of the data center. In [18], the authors propose an ant colony system (ACS) based algorithm for VM placement whose main goals are minimizing the energy consumption and resource wastage. Similar to this work, the authors of [19] present a biogeography-based optimization method to achieve the aforementioned goals. A multi-objective ant colony optimization (MACO) approach to host VMs on PMs with the aim of minimizing energy consumption, resource wastage and communication energy cost is proposed in [24]. Also, the authors of [15] introduce an energy-aware method based on the cultural algorithm for placing VMs on PMs. In [25], the authors propose a deep reinforcement learning approach for resource provisioning and task scheduling to minimize the energy cost of CPs. To balance the load in PMs, the authors of [26] present a reinforcement learning based algorithm to efficiently replace VMs.

In [27], the authors focus on the online VM consolidation problem and propose a micro genetic-based approach for minimizing the energy consumption of a CDC. The authors of [28], take into account the maximization of VM performance and the balancing of PM workload in their optimization problem and then solve it using a greedy-based algorithm. The authors in [29] propose a thermal-aware scheduling method to minimize the temperature of a CDC. The simulation results show that their strategy can improve the energy consumption and other QoS parameters such as latency and temperature. In [20], we formulate the VM placement as a bi-objective problem and present an efficient greedy randomized approach based on the combination of the 'power of two choices' model [30] and the first fit strategy [31] to minimize the energy consumption and resource wastage in a data center. It is worth to mention that the main differences between this work compared to our previous work are as follows. First, in our previous work, i.e., [20], we have not considered the data center network (DCN) topology and the traffic matrix among VMs while in this study we consider both of them. Second, in that work, we have assumed that all requests have the same priority while the present paper is priority-aware, i.e., requests can have different priority.

Although the above-mentioned algorithms are promising solutions for VM placement, they have not considered the impact of communication traffic between VMs on a data center network.

### 2.2. Network-aware VMP

Traffic communication between VMs has drastic impact on the overall performance of a DCN. Hence, in the literature, many approaches take into account this aspect and propose network-aware VM placement. In [21], the authors present a traffic-aware VMP problem (TVMPP) to improve the network consumption of a data center. To achieve this objective, they propose a two-tier heuristic algorithm which places VMs with large mutual traffic on PMs with low-cost connections. The authors in [32] propose a graph community-based algorithm for mapping VMs to PMs in DCNs. The algorithm first makes partitions of PMs regrading to their connectivity, then clusters VMs based on the amount of traffic exchanged between them. After that, it uses the well-known bin packing strategies to fit the clusters in the partitions. EQVMP is an energy-efficient and QoS-aware approach for VM placement in software defined DCNs [22]. EQVMP is a three-tier algorithm; First, it partitions VMs into uniform groups with low traffic connection among them. Then, it inspires a heuristic-based bin packing algorithm to place VMs on PMs to reduce the number of power-on PMs. After the placement process, a load balancing mechanism is used to improve the network performance. An ACO based approach for VM placement with the aim of minimizing energy consumption is proposed in [33]. The authors consider both PM-side and network-side constraints in their optimization formulation.

The authors of [34] present a topology-aware algorithm to consolidate groups of VMs in a small area of a hierarchical DCN. For each group of VMs, the algorithm selects an area with the least increasing of power consumption. The authors also consider the path allocation for the flows between VMs. Another topology-aware VM placement algorithm is proposed in [35] which aims to minimize the maximum link utilization of a DCN. The basic idea of this work is placing all the VMs of a request on a PM as far as possible; otherwise, the request is split into several sub-requests. Then, sub-requests are sorted based on their number of VMs is ascending order and best fit strategy is used to host VMs. PAVA is a priority-aware VM allocation approach in SDN-enabled clouds [4]. PAVA tries to place VMs of the high-priority application on the host group having more computing and networking resources. For a low-priority application, the authors use first fit decreasing (FFD) strategy. They also propose a bandwidth allocation (BWA) method to allocate the required network bandwidth for a high-priority application. Recently, the authors in

[23] present a three-objective VM placement optimization problem to minimize power consumption, resource wastage and bandwidth usage and propose a hybrid multi-objective genetic algorithm for solving the problem.

Table 1 summarizes existing works and highlights the research gap that is covered by this work. To the best of our knowledge, this is the first work that considers the energy consumption, resource wastage and network consumption of a DCN, the heterogeneity of PMs, and the priority of application requests altogether.

Table 1: Summary of literature review. App:=Applications; Homo:=Homogeneous; Hetero:= Heterogeneous.

| References | Energy-aware | Resource-aware | Network-aware | App Priority | PM Type | Approach |
|---|---|---|---|---|---|---|
| Beloglazov et al. [13] | ● | ○ | ○ | ○ | Hetero | Heuristic |
| Gao et al. [18] | ● | ● | ○ | ○ | Hetero | ACS |
| Zheng et al. [19] | ● | ● | ○ | ○ | Hetero | Biogeography |
| Malekloo et al. [24] | ● | ● | ○ | ○ | Hetero | ACO |
| Mohammadhosseini et al. [15] | ● | ○ | ○ | ○ | Hetero | Cultural |
| Cheng et al. [25] | ● | ○ | ○ | ● | Hetero | Deep Reinforcement Learning |
| Ghasemi et al. [26] | ● | ● | ○ | ○ | Hetero | Reinforcement Learning |
| Tarahomi et al. [27] | ● | ● | ○ | ○ | Hetero | Micro-genetic |
| Zhao et al. [28] | ○ | ● | ○ | ○ | Homo | Greedy |
| Tuli et al. [29] | ● | ○ | ○ | ○ | Hetero | Artificial Intelligence |
| Azizi et al. [20] | ● | ● | ○ | ○ | Hetero | Greedy Randomized |
| Meng et al. [21] | ○ | ○ | ● | ○ | Homo | Heuristic |
| Dias et al. [32] | ○ | ○ | ● | ○ | Homo | Heuristic |
| Wang et al. [22] | ● | ○ | ● | ○ | Hetero | Heuristic |
| Gao et al. [33] | ● | ○ | ● | ○ | Hetero | ACO |
| da Silva et al. [34] | ● | ○ | ● | ○ | Hetero | Heuristic |
| Li et al. [35] | ○ | ○ | ● | ○ | Homo | Heuristic |
| Son et al. [4] | ● | ○ | ● | ● | Hetero | Heuristic |
| Farzai et al. [23] | ● | ● | ● | ○ | Hetero | GA |
| **This Study** | ● | ● | ● | ● | **Hetero** | **Heuristic** |

## 3. Proposed Architecture

In this section, we first describe the considered system architecture which is illustrated in Fig. 1. Specifically, we discuss about its components along with interaction between them. Then, we present an ILP model for the virtual machine placement problem.

Cloud users submit their application requests to the cloud provider through the Internet. Each application request is characterized by some specific profiles including its priority information, number of VMs, flows between VMs, resource requirements of each VM, and so on. These specifications can be set based on user requirements which is provided by the most of cloud service providers such as Amazon AWS an Microsoft Azure [4]. It is worth mentioning that application requests can be submitted to the system at any time.

On the cloud provider side, there exist three main components: named Application Receiver, Resource Manager and Allocation Manager. Fig. 1 shows these components along with the interaction between them. In the following, we describe the function of each of these components in detail.

The application receiver component is responsible for receiving application requests from end users. When an application request is submitted to this component, it analyzes the resource requirement of the application VMs. Then, it asks the resource manager component whether PMs have enough resources to host this application or not. If the answer is 'yes', it accepts the request and pass it to the allocation manager component to find a suitable place to host the VMs of the application. Otherwise, it rejects the request and informs the user. In the allocation manager, the priority analyzer module specifies the priority of the requested application, which can be determined by either the user of the cloud provider [4]. In this work, similar to [4], we consider two priorities for applications: *critical* (high priority) and *normal* (low priority). Based on the output of the priority analyzer module, one of our proposed algorithms is executed. If the requested application is critical, our JointPT is running to find a suitable VM-to-PM mapping; otherwise, our JointPR is running. Our algorithms solve the VMP problem based on the information provided by the resource manager component (such as the list of available PMs, their remaining resource capacity, their power consumption profile, and so on). Finally, the offered solution is sent to VM placer module to host VMs on the relevant PMs.

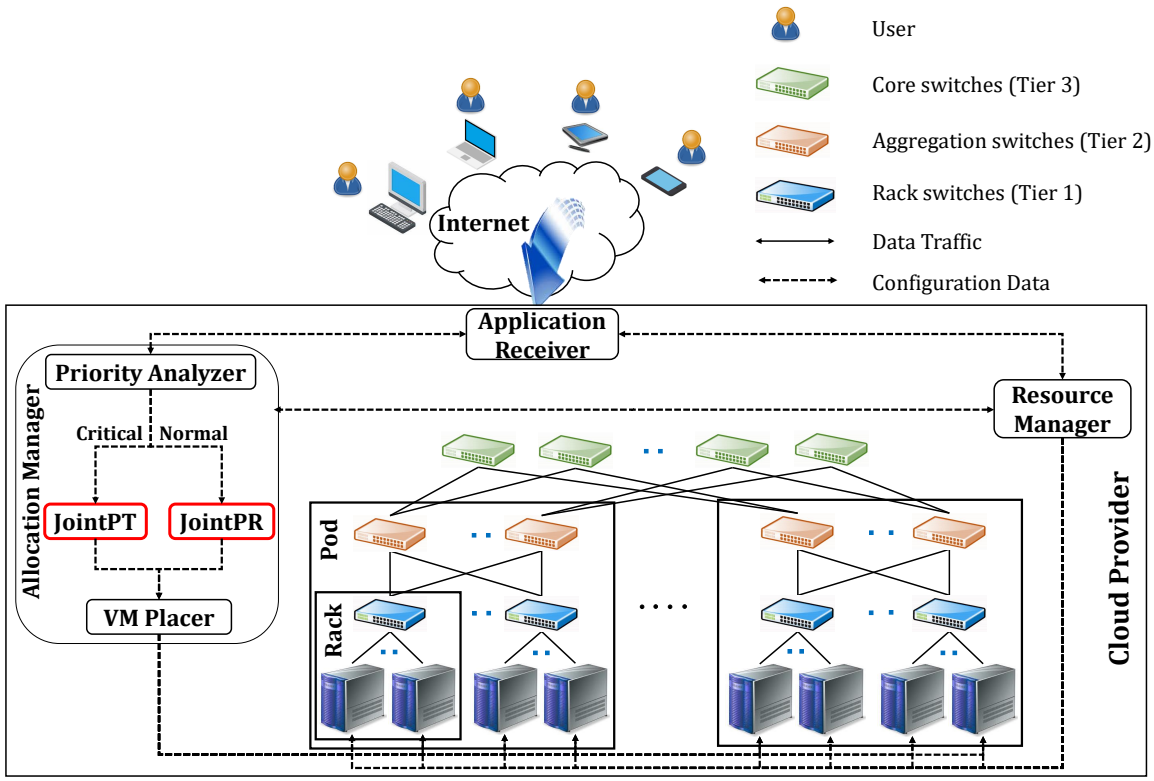


Figure 1: The considered system architecture for the provider of Cloud Data Center (CDC) where it consists of rack; Pods connecting to the Allocation Manager. JointPT:= Joint Power and Traffic optimization algorithm; JointPR:= Joint Power and Resource optimization algorithm.

The computing resources in our system architecture are PMs connecting to each other based on a particular physical topology. In this work, similar to many works [22, 4, 36, 16, 23], we consider the well-known, three-tier fat-tree topology [37] as depicted in Fig. 1. At the tier 1, there are $r/2$ servers connected to a $r$-port Rack switch. At the tier 2, $r/2$ of Racks are connected together through $r/2$ aggregation switches which constitute a Pod. Finally, at last tier, $r$ Pods are connected to each other through $(r/2)^2$ core switches.

Table 2: Main notation.

| | Symbol | Definition | Type - Unit | Appears in Eq. |
|---|---|---|---|---|
| **Set** | $A$ | Set of applications and $\mid A \mid = M$ | - | - |
| | $V$ | Set of of VMs and $\mid V \mid = n$ | - | - |
| | $E$ | Set of edges as dependencies among VMs of an application and $\mid E \mid = E^{max}$ | - | - |
| | $P$ | Set of PMs and $\mid P \mid = N$ | - | - |
| **Index** | $k, k^{'}$ | Index of PM, $k, k^{'} \in P$ | Integer - [units] | - |
| | $i, j$ | Index of VM, $i, j \in V$ | Integer - [units] | - |
| | $l$ | Index of application , $l \in A$ | Integer - [units] | - |
| **Input Parameters** | $r$ | Number of switch ports | Integer - [units] | - |
| | $e_{ij}$ | Amount of traffic between VMs $V_i$ and $V_j$ | Continues - [units] | (10) |
| | $P_k^c$ | CPU capacity of $P_k$ | Integer - [cores] | (3), (5), (13) |
| | $P_k^m$ | Memory capacity of $P_k$ | Continues - [GB] | (4), (14) |
| | $V_i^c$ | CPU demand of $V_i$ | Integer - [cores] | (3), (13) |
| | $V_i^m$ | Memory demand of $V_i$ | Continues - [GB] | (4), (14) |
| | $P_k^{full}$ | Power consumption of $P_k$ in full utilization state | Continuous - [watt] | (5), (6) |
| | $P_k^{idle}$ | Power consumption of $P_k$ in idle state | Continuous - [watt] | (6) |
| | $U_k^c$ | Normalized CPU utilization of $P_k$ | Continuous - [units] | (3), (6), (8) |
| | $U_k^m$ | Normalized memory utilization of $P_k$ | Continuous - [units] | (4), (8) |
| | $R_k^c$ | Normalized remaining CPU utilization of $P_k$ | Continuous - [units] | (8) |
| | $R_k^m$ | Normalized remaining memory utilization of $P_k$ | Continuous - [units] | (8) |
| | $\mathscr{P}_k$ | Power efficiency of $P_k$ | Continuous - [units] | (8) |
| | $P_k^{power}$ | Power consumption of $P_k$ | Continuous - [watt] | (6), (7) |
| | $R_k^{wastage}$ | Resource wastage of $P_k$ | Continuous - [watt] | (8), (9) |
| | $N_l^{network}$ | Network consumption for the VMs of application $A_l$ | Continuous - [units] | (10), (11) |
| | $D(V_i, V_j)$ | Shortest distance between the source and destination PMs hosted $V_i$ and $V_j$ | Integers - [hops] | (10) |
| | $\mathscr{U}_{ik}^c$ | Normalized CPU utilization of $P_k$ after placing $V_i$ | Continuous - [units] | (17) |
| | $\mathscr{U}_{ik}^m$ | Normalized memory utilization of $P_k$ after placing $V_i$ | Continuous - [units] | (17) |
| | $\mathscr{R}_{ik}^c$ | Normalized remaining CPU utilization of $P_k$ after placing $V_i$ | Continuous - [units] | (17) |
| | $\mathscr{R}_{ik}^m$ | Normalized remaining memory utilization of $P_k$ after placing $V_i$ | Continuous - [units] | (17) |
| | $\mathscr{R}_{ik}^{wastage}$ | Resource wastage of $P_k$ after placing $V_i$ | Continuous - [units] | (17) |
| | $\varepsilon$ | Very small positive real number | Continuous - [units] | (8), (17) |
| **Variables** | $x_{ik}$ | Binary variable to show if VM $V_i$ is placed on PM $P_k$ | Binary - [units] | (1), (3), (4), (13)-(16) |
| | $y_k$ | Binary variable to show if PM $P_k$ is active | Binary - [units] | (2), (7), (9), (13), (14), (16) |
| | $\mathbb{P}^{tot}$ | Total power consumption of a CDC | Continuous - [watt] | (7), (12) |
| | $\mathbb{R}^{tot}$ | Total resource wastage of a CDC | Continuous - [units] | (9), (12) |
| | $\mathbb{N}^{tot}$ | Total network consumption of a CDC | Continuous - [units] | (11), (12) |

## 4. Optimization Problem and Formulation

In this section, we present a mathematical model of the VMP optimization regarding to the proposed system architecture. For easy reference, all notation used throughout the paper is summarized in Table 2.

### *4.1. List of Applications*

Suppose $A = \{A_1, A_2, \ldots, A_M\}$ is the set of $M$ applications that are submitted to the cloud provider by the cloud users, one by one. Formally, an application can be modeled as a graph $G = (V, E)$, where $V = \{V_1, V_2, \ldots, V_n\}$ is the set of VMs and $E$ is the set of edges representing dependencies among VMs. Each edge $e_{ij} = (V_i, V_j)$ indicates the amount of traffic between VMs $V_i$ and $V_j$. We denote the size of $E$ as $E^{max}$, i.e., $|E| = E^{max}$. A VM has some specific resource requirements such as the number of processing cores, the amount of memory, the size of storage, and so on. In this work, we focus on the processing power and memory, which are the most commonly used characteristics of VMs [18, 38, 23]. We use notations $V_i^c$ and $V_i^m$ to represent respectively the number of CPU cores and the amount of memory needed by VM $V_i$, for all $i \in \{1, 2, \ldots, n\}$. We should mention that each application can consist of any number of VMs.

*4.2. List of Physical Machines (PMs)*

In a cloud data center with a three-tier fat-tree network topology and $r$-port switches, $N = r^3/4$ PMs can be networked together. For example, using 16-port switches, we will have a data center network including 1024 PMs. So let $P = \{P_1, P_2, \ldots, P_N\}$ be a set of $N$ heterogeneous PMs in the data center in which the capacity of each PM in terms of the number of CPU cores and the amount of memory is denoted by $P_k^c$ and $P_k^m$, respectively, for all $k \in \{1, 2, \ldots, N\}$. We also use notations $P_k^{full}$ and $P_k^{idle}$ to represent respectively the power consumption profile of PM $P_k$ in the full utilization and idle state, for all $k \in \{1, 2, \ldots, N\}$.

*4.3. Optimization model*

The main goal of the VMP is finding an optimized solution for mapping function $f : V \rightarrow P$ in a way that some predefined objectives and constraints are met. In the following, we present a MILP model of the addressed problem. The main decision variable of our optimization problem is a binary variable which can be defined as follows:

$$x_{ik} = \begin{cases} 1 & \text{if VM } V_i \text{ is hosted to PM } P_k \\ 0 & \text{otherwise} \end{cases}, \quad \forall V_i \in \boldsymbol{V}, \forall P_k \in \boldsymbol{P} \tag{1}$$

We say that a PM is active if at least one VM is placed on it; otherwise, it is inactive. To mathematically identify the state of a PM, we use another decision variable which can be represented by the following equation:

$$y_k = \begin{cases} 1 & \text{if } \sum_{i=1}^{n} x_{ik} \geq 1 \\ 0 & \text{otherwise} \end{cases}, \quad \forall P_k \in \boldsymbol{P} \tag{2}$$

For an active PM $P_k$, we define $U_k^c$ and $U_k^m$ to specify its normalized CPU and memory utilization, respectively, which are obtained using the following equations.

$$U_k^c = \sum_{l=1}^{M} \sum_{i=1}^{n} \frac{x_{ik} \times V_i^c}{P_k^c}, \quad \forall P_k \in P \tag{3}$$

$$U_k^m = \sum_{l=1}^{M} \sum_{i=1}^{n} \frac{x_{ik} \times V_i^m}{P_k^m}, \quad \forall P_k \in P \tag{4}$$

These values can be obtained by dividing the resource requirements of all VMs placed on that PM by its resource capacity. We also use notations $R_k^c = (1 - U_k^c)$ and $R_k^m = (1 - U_k^m)$ to express the normalized remaining CPU and memory of PM $P_k$, respectively. It is worth to mention that in our considered system architecture, the value of these parameters can be obtained by the resource manager component (see Fig. 1).

In a cloud data center with heterogeneous computing resources, PMs usually have different power consumption and computing power characteristics. Therefore, we can define a key parameter to identify the efficiency of a PM. In this regard, we define the power efficiency of PM $P_k$ as follows [39]:

$$\mathscr{P}_k = \frac{P_k^c}{P_k^{full}}, \quad \forall P_k \in P \tag{5}$$

The objectives of the current study are to place the VMs of applications in a way that power consumption, resource wastage and network consumption are minimized. Hence, in the following, we give a mathematical model for each of these objectives.

*4.3.1. Power consumption model*

The power consumption of a PM highly depends on its CPU utilization. The most common model used for calculating the power consumption of a PM is a linear model presented in [40]. Accordingly, we use the following equation to calculate the power consumption of PM $P_k$.

$$P_k^{power} = \begin{cases} P_k^{idle} + \left(P_k^{full} - P_k^{idle}\right) \times U_k^c & \text{if } U_k^c > 0 \\ 0 & \text{otherwise} \end{cases}, \quad \forall P_k \in P \tag{6}$$

Based on the above formula, the total power consumption of a CDC can be obtained using the below equation:

$$\mathbb{P}^{total} = \sum_{k=1}^{N} y_k \times P_k^{power} \tag{7}$$

*4.3.2. Resource wastage model*

To fully utilize of a multi-dimensional, activated PM, its resource utilization must be maximized and balanced in all dimensions. In other words, its resource wastage must be minimized. To this end, in [18], authors have proposed an intelligent equation which calculates the value of this parameter for a PM $P_k$ as the following

$$R_k^{wastage} = \frac{\mid R_k^c - R_k^m \mid + \varepsilon}{U_k^c + U_k^m}, \quad \forall P_k \in P \tag{8}$$

where $\varepsilon$ is a very small positive real number that we set it to 0.0001, similar to [18].

Having this equation, the total resources wasted by a VM placement algorithm is obtained using the following equation:

$$\mathbb{R}^{total} = \sum_{k=1}^{N} y_k \times R_k^{wastage} \tag{9}$$

*4.3.3. Network consumption model*

To reduce the communication cost among the VMs of an application during the execution, they must be carefully placed. To determine how a placement algorithm achieves this goal, we need to define an appropriate metric based on the distances between the source and destination PMs of all VMs of an application. In this work, we introduce a metric called network consumption to measure the proximity of an application VMs in a fat-tree network topology. The proposed metric is defined based on the application graph (or traffic matrix) and the location of PMs in the topology.

For each edge $e_{ij} = (V_i, V_j)$ of the application graph, let $P_k$ and $P_{k'}$ be the source and the destination PMs of $V_i$ and $V_j$, respectively. We define $D(V_i, V_j)$ as the shortest distance in the number of hops between their source and destination PMs. Regarding the topological structure of fat-tree, there are four possible cases for two PMs $P_k$ and $P_{k'}$:
*Case 1:* They are the same host. In this case, $D(V_i, V_j) = 0$; because $V_i$ and $V_j$ can communicate with each other through the host memory without injecting any traffic to the network.
*Case 2:* $P_k$ and $P_{k'}$ are within the same Rack. So the traffic between two VMs is transferred through a Rack switch which needs two hops, i.e., $D(V_i, V_j) = 2$.
*Case 3:* Two PMs are inside the same Pod, but with a different Rack. In this case, each packet must traverse four hops from the source VM to the destination VM, i.e., $D(V_i, V_j) = 4$.
*Case 4:* The last case occurs when VMs $V_i$ and $V_j$ are hosted on the PMs located within the different Pods, where they will be six hops apart from each other. Thus, $D(V_i, V_j) = 6$.

Hence, the network consumption for the VMs of an application, say $A_l$, is defined as follows:

$$N_l^{network} = \sum_{i,j \in V} D(V_i, V_j) \times e_{ij}, \quad \forall l \in A \tag{10}$$

The key idea behind this equation is to place mutual VMs with the higher traffic demand on PMs with the closest proximity. This leads to a reduction in traffic injected to a DCN. The total network consumption of a CDC for $M$ application requests can be obtained using the following equation:

$$\mathbb{N}^{total} = \sum_{l=1}^{M} N_l^{network} \tag{11}$$

*4.3.4. Objective function*

The objective function of the considered VMP problem can be formulated as the following an MILP model:

$$\min \quad (\mathbb{P}^{total} + \mathbb{R}^{total} + \mathbb{N}^{total}) \tag{12}$$

s.t:

$$\sum_{l=1}^{M}\sum_{i=1}^{n} x_{ik} \times V_i^c \leq y_k \times P_k^c, \quad \forall P_k \in P \tag{13}$$

$$\sum_{l=1}^{M}\sum_{i=1}^{n} x_{ik} \times V_i^m \leq y_k \times P_k^m, \quad \forall P_k \in P \tag{14}$$

$$\sum_{k=1}^{N} x_{ik} = 1, \quad \forall V_i \in V \tag{15}$$

$$x_{ik}, y_k \in 0,1, \quad \forall i \in V, k \in P \tag{16}$$

where constraints (13) and (14) respectively ensure that the CPU and memory utilization of active PMs must not exceed their capacity. Constraint (15) indicates that each VM can be hosted only on one PM. Constraint (16) determines that the decision variables used in the model are binary.

Since VMP is an NP-hard problem [41], proposing an efficient exact algorithm is unlikely to exist. Considering multi-dimensionality, heterogeneity, and multi-objectivity make it even harder. Therefore, in this work, we present two efficient heuristic algorithms to cope with this problem.

## 5. Proposed heuristic algorithms

In this section, we explain the proposed heuristic algorithms to solve the VMP problem efficiently. For critical applications, we present JointPT whose major intuitions are *twofold*: i) minimizing the consumption of power by exploiting the power-efficient PMs and reducing the number of active PMs, and ii) reducing the consumption of network by considering the traffic between VMs and hosting them on PMs with the closest proximity. For normal ones, we present JointPR, which aims to minimize both power consumption and resource wastage in a CDC. To achieve the former goal, JointPR uses a strategy similar to JointPT; while for the next goal, it tries to find a suitable VM-PM mapping to maximize resource utilization and balancing the load among multi-dimensional resources of activated PMs. The detail of the proposed algorithms is discussed in the following subsections.

### *5.1. JointPT*

The JointPT algorithm takes the network topology of PMs, inter-VM traffic matrix, resource and power consumption profile of PMs, and resource demand of VMs as input. Then, it performs three major phases as follows.

*Phase-1:* In the first phase, JointPT calculates the power efficiency of each PM (based on eq.(5)) and then sorts Pods, the Racks inside each Pod, and PMs within each rack in descending order of their power efficiency. It means the algorithm gives higher priority to the power-efficient PMs.

*Phase-2:* In the second phase, JointPT determines the order of a requested application VMs according to the Maximum Spanning Tree (MST) algorithm. Such a tree can be easily found using prim's algorithm. For this end, we first multiply the edge weights by -1 and then solve the minimum spanning tree problem on the obtained graph. An illustrative example is shown in Fig. 2. Let us assume that VM $V_2$ is the starting node. The sequence order of VMs is as follows: $V_2, V_5, V_1, V_3, V_4, V_6$.

*Phase-3:* After determining the sequence order of VMs, the JointPT algorithm tries to allocate mutual VMs with the higher traffic demand into PMs with the closest proximity. To this end, in the first attempt, JointPT will try to host VMs with high traffic intensity on the same PM. If it could not do this, PMs located on the same Rack and then the same Pod is examined. In the last attempt, a PM outside the Pod is found.

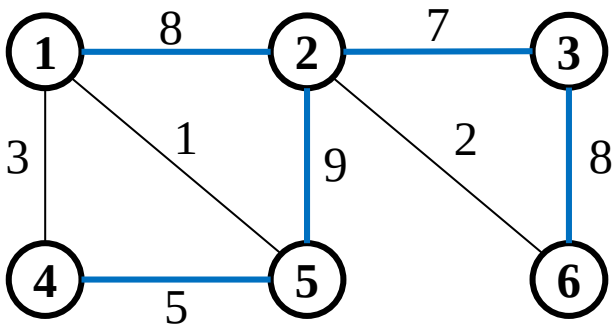


Figure 2: Maximum spanning tree for a traffic matrix among VMs

**Algorithm 1** presents the pseudo-code of the proposed JointPT. First, the algorithm calculates the power efficiency of PMs (line 1). After that, it sums up the power efficiency value of PMs inside each Pod (and each Rack), and then sort Pods (and Racks) in descending order based on the obtained values (lines 2 and 3). The algorithm also sorts PMs inside each Rack based on their power efficiency (line 4). In line 5, the sequence order of VMs is calculated using Prim's algorithm for the maximum spanning tree. Next, the algorithm iterates through VMs to host them on suitable PMs (lines 6-33). Line 7 checks if $V_i$ is the first VM of the requested application; If so, JointPT uses the First Fit (FF) heuristic algorithm to allocate it on the first PM with sufficient capacity (line 8); otherwise, our algorithm will attempt to find a PM which has enough resources to host $V_i$ and provides the minimum network distance from the previous VM, say $V_{i-1}$ (line 9 to 32).

**Algorithm 1** Joint Power and Traffic optimization algorithm (JointPT)

**Input:** $G(V,E)$: Application Graph, $P$: List of all PMs in DC
**Output:** VMP map

1: **calculate** the power efficiency of PMs using eq. (5);
2: **sort** Pods based on their power efficiency in descending order;
3: **sort** Racks inside each Pod based on their power efficiency in descending order;
4: **sort** $P_k$ inside each Rack based on their power efficiency in descending order;
5: **calculate** the sequence order of VMs according to the MST algorithm;
6: **for each** $V_i \in \boldsymbol{V}$ **do**
7: **if** $i == 1$ **then** ▷ $V_i$ is the first VM for placement
8: **use** FF algorithm to place $V_i$;
9: **else**
10: $P_i \leftarrow$ a PM that $V_{i-1}$ is hosted on it;
11: Q.insert($P_k$);
12: $Rack_i \leftarrow$ a Rack that $V_{i-1}$ is included;
13: **for each** $P_k \in R_i$ **do**
14: Q.insert($P_k$);
15: **end for**
16: $Pod_i \leftarrow$ a Pod that $V_{i-1}$ is included;
17: **for each** $P_k \in Pod_i$ **do**
18: Q.insert($P_k$);
19: **end for**
20: **for each** $P_k \in$ DC **do**
21: Q.insert($P_k$);
22: **end for**
23: **while** $(Q \neq \emptyset)$ **do**
24: $P_k \leftarrow$ delete(Q);
25: **if** $V_i^c \leq P_k^c$ and $V_i^m \leq P_k^m$ **then**
26: **place** $V_i$ on $P_k$;
27: $P_k^c \leftarrow P_k^c$ - $V_i^c$;
28: $P_k^m \leftarrow P_k^m$ - $V_i^m$;
29: **break;**
30: **end if**
31: **end while**
32: **end if**
33: **end for**

### *5.2. JointPR*

The major concerns in the JointPR algorithm are minimizing the power consumption and maximizing the utilization of the activated PMs. To achieve these goals, the algorithm performs two main phases. The first one is the same as the *Phase-1* of JointPT. For the second phase, however, JointPR maps VMs on PMs in a way that the resource wastage of the active PMs is minimized. Thus, for each VM, the algorithm checks all the active PMs and place it on the PM with sufficient resources and the least resource wastage. To calculate the resource wastage of PM $P_k$ after mapping VM $V_i$, we can use the following equation:

$$\mathscr{R}_{ik}^{wastage} = \frac{|\mathscr{R}_{ik}^c - \mathscr{R}_{ik}^m)| + \varepsilon}{\mathscr{U}_{ik}^c + \mathscr{U}_{ik}^m}, \quad i \in V, k \in P \tag{17}$$

where $\mathscr{R}_{ik}^c$ and $\mathscr{R}_{ik}^m$ are the normalized remaining CPU and memory of PM $P_k$ if VM $V_i$ is located on it, respectively. We can use the same description for $\mathscr{U}_{ik}^c$ and $\mathscr{U}_{ik}^m$ which are normalized CPU and memory utilization of PM $P_k$.

**Algorithm 2** shows the pseudo-code of the JointPR algorithm. Lines 1 to 4 are similar to **Algorithm 1**. In line 5, the algorithm creates a list from the previously activated PMs and puts them in the set P. After that the proposed algorithm runs the outer loop (lines 6 to 27) to map application VMs on the most suitable PMs. To do this, for each VM $V_i$, it searches among all the active PMs and tries to find a PM with the sufficient capacity and the least resource wastage value, after placing VM $V_i$ (see lines 7 to 16). If such a PM is available, the algorithm places the VM on that PM and updates its resources (lines 17 to 20); otherwise, the next power efficient PM is activated to host the VM (lines 21 to 26). It is worth mentioning that here we assume that an activated PM can host at least one VM.

**Algorithm 2** Joint Power and Resource optimization algorithm (JointPR)

**Input:** $G(V,E)$: Application Graph, $P$: List of all PMs in DC
**Output:** VMP map

**calculate** the power efficiency of PMs using eq. (5);
**sort** Pods based on their power efficiency in descending order;
**sort** Racks inside each Pod based on their power efficiency in descending order;
**sort** $P_k$ inside each Rack based on their power efficiency in descending order;
$\mathscr{P} \leftarrow$ list of active PMs;
**for each** $V_i \in V$ **do**
    **for each** $P_k \in \mathscr{P}$ **do**
        min $\leftarrow \infty$
        **if** $V_i^c \leq P_k^c$ and $V_i^m \leq P_k^m$ **then**
            **calculate** $R_{ik}^{wastage}$ using eq.(17)
            **if** $R_{ik}^{wastage} < min$ **then**
                min $\leftarrow R_{ik}^{wastage}$;
                index $\leftarrow$ k;
            **end if**
        **end if**
    **end for**
    **if** $min \neq \infty$ **then**
        **place** $V_i$ on $P_{index}$;
        $P_{index}^c \leftarrow P_{index}^c$ - $V_i^c$;
        $P_{index}^m \leftarrow P_{index}^m$ - $V_i^m$;
    **else**
        **activate** the next PM $P_{k'}$ from the sorted list $P$ and **add** it to $\mathscr{P}$;
        **place** $V_i$ on $P_{k'}$;
        $P_{k'}^c \leftarrow P_{k'}^c$ - $V_i^c$;
        $P_{k'}^m \leftarrow P_{k'}^m$ - $V_i^m$;
    **end if**
**end for**

To make our proposed method easy to understand, Fig.3 provides its flowchart description.

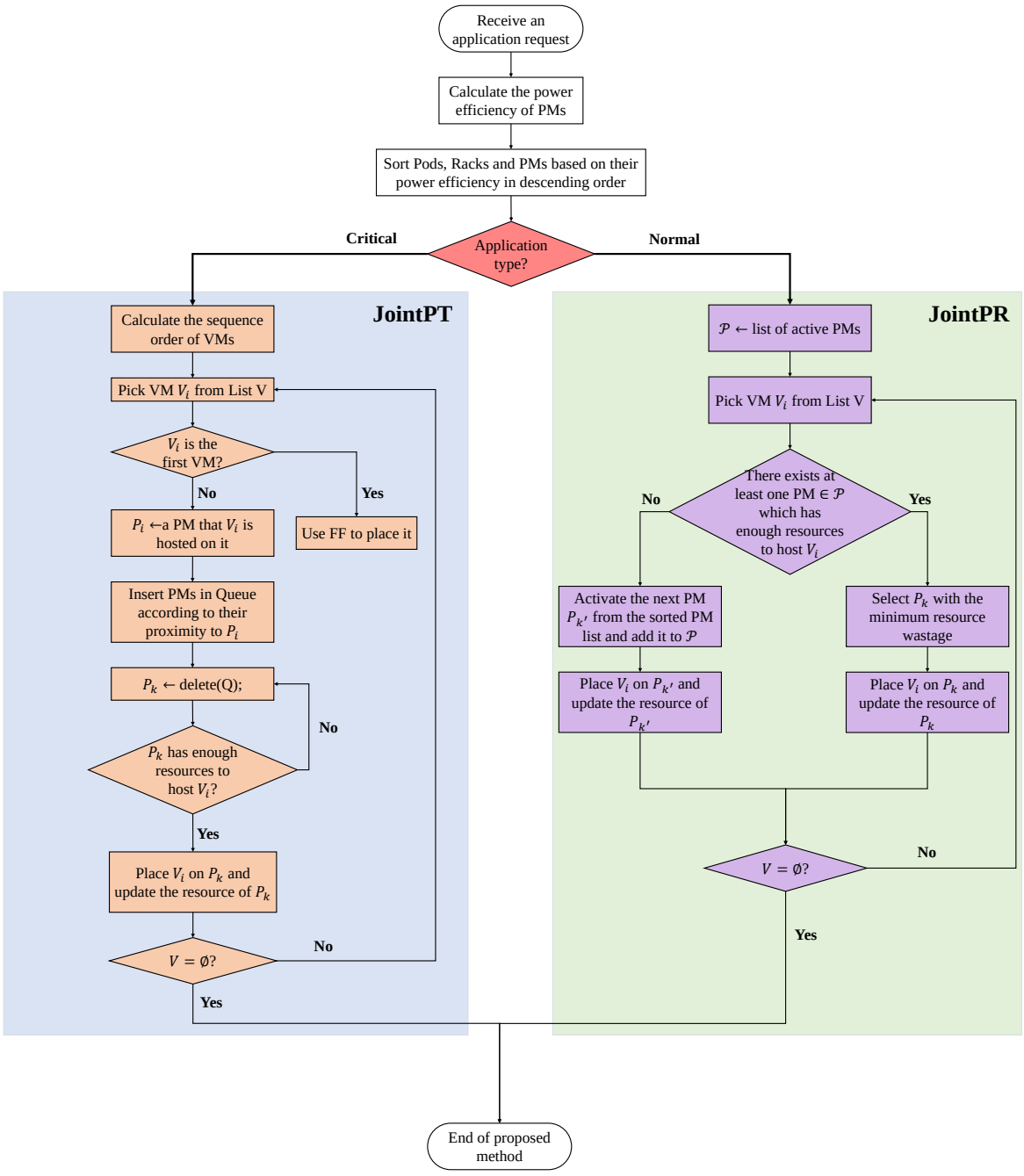


Figure 3: Flowchart of the proposed method

### *5.3. Complexity Analysis*

Here, we present the time complexity of the proposed algorithms. Remember that a fat-tree topology with $r$-port switches consists of $N = r^3/4$ PMs, $r$ Pods, $r/2$ Racks inside each Pod, and $r/2$ PMs connected to each Rack. Also,

$n$ represents the number of VMs of a requested application.

***JointPT:*** The complexity of line 1 is $O(N)$. Lines 2, 3 and 4 requires $O(r \times \log r)$, $O(r^2/2 \times \log(r/2))$ and $O(N \times \log(r/2))$, respectively. Calculating the MST using Prime's algorithm (line 5) requires $O(E^{max} \times \log n)$. The complexity analysis of lines 6 to 33 is as follows. For the first VM, the algorithm uses the FF algorithm (lines 7 and 8) wherein the worst case it should examine all PMs, i.e., $O(N)$. For other $(n-1)$ VMs, the algorithm first insert all PMs to a queue (lines 9 to 22) and then delete them from the queue (lines 23 to 32). Since we use a simple First In First Out (FIFO) queue, both of them requires $O(N)$. Thus, the complexity of lines 6 to 33 in the worst case is equal to $O(n \times N)$. Therefore, the overall time complexity of JointPT for a requested application is $O(N \times \log(r/2) + E^{max} \times \log n + n \times N)$.

***JointPR:*** The complexity of lines 1 to 4 is similar to the above analysis. In line 5, the algorithm needs the list of active PMs, which requires $O(N)$. The rest of the algorithm includes two nested loops (lines 6 to 27). The first loop is executed for all $n$ VMs while the second one requires to check at most $N$ PMs. Hence, the overall time complexity of JointPR is equal to $O(N \times \log(r/2) + n \times N)$.

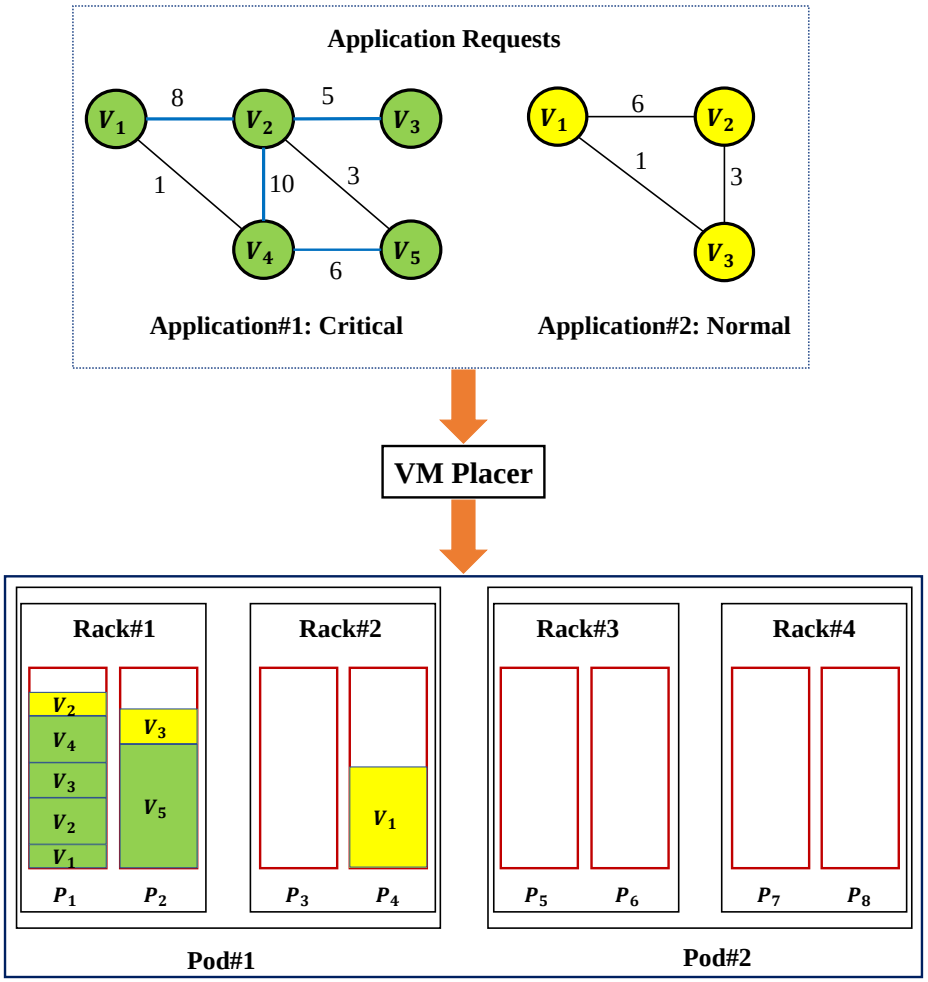


Figure 4: An illustrative example

*5.4. An Illustrative Example*

To better understand of the proposed heuristic algorithms, here we give an illustrative example. For this end, we consider a fat-tree topology with two available Pods, two Racks inside each Pod, and two PMs within each Rack. We also consider two different application requests, one critical and one normal (see Fig.4). The characteristics of PMs and VMs of applications are described in Tables 3, 4 and 5, respectively. Let us assume that Application#1 enters the system first. We follow the next steps to map VMs to PMs based on the proposed JointPT and JointPR algorithms.
**Step 1:** The power efficiency of the PMs is calculated using eq. (7) as in Table 3.
**Step 2:** Based on the sorting phase of JointPT, the priority of PMs is obtained, where Pod#1 has higher priority compared to Pod#2, Rack#1 has higher priority in comparison with Rack#2, $P_1$ is more power efficient than $P_2$, and so on.
**Step 3:** The sequence order of Application#1 VMs is as follows: $V_1$, $V_2$, $V_4$, $V_5$, and $V_3$.
**Step 4:** By checking the capacity of PMs and requirement of VMs, the placement is done, as it can be seen in Fig. 4.
**Step 5:** For Application#2, JointPR first tries to place $V_1$ on active PMs, i.e., $P_1$ and $P_2$. However, these PMs cannot host $V_1$. Therefore, this VM is located on the next power efficient PM with sufficient resources which is $P_4$. The other two VMs, $V_2$ and $V_3$, are placed on the PMs with the minimum resource wastage that is $P_1$ and $P_2$, respectively.

Table 3: PMs' Characteristics

| #PM | CPU [cores] | Memory [GB] | $P^{full}$ [W] | $\mathscr{P}$ |
|---|---|---|---|---|
| $P_1$ | 64 | 128 | 186 | 0.344 |
| $P_2$ | 24 | 64 | 112 | 0.214 |
| $P_3$ | 24 | 32 | 148 | 0.162 |
| $P_4$ | 32 | 64 | 156 | 0.205 |
| $P_5$ | 24 | 48 | 164 | 0.146 |
| $P_6$ | 16 | 32 | 132 | 0.121 |
| $P_7$ | 24 | 32 | 148 | 0.162 |
| $P_8$ | 32 | 48 | 196 | 0.163 |

Table 4: VMs' Characteristics (Application#1).

| #VM | vCPU [cores] | Memory [GB] |
|---|---|---|
| $V_1$ | 8 | 16 |
| $V_2$ | 16 | 64 |
| $V_3$ | 12 | 16 |
| $V_4$ | 16 | 16 |
| $V_5$ | 16 | 32 |

Table 5: VMs' Characteristics (Application#2).

| #VM | vCPU [cores] | Memory [GB] |
|---|---|---|
| $V_1$ | 16 | 32 |
| $V_2$ | 8 | 4 |
| $V_3$ | 4 | 16 |

## 6. Performance Evaluation

To evaluate the performance of the proposed method, we first describe simulation settings. After that benchmark algorithms are introduced. Finally, we present results. The source code of our paper is available in [42].

*6.1. Simulation setup*

For conducting the experiments, we used Java programming language. The experiments were carried out on a PC with Intel Core i7-6400U CPU 2.10 GHz (4 processors), 12 GB RAM, and Windows 10 OS. To provide results with high confidence, each experiment is executed 10 times and reported an average of them. In order to examine the impact of parameters, we tacked into consideration three various experiments. For experiment one, we varied the number of applications from 4, 8, 16, 32, 64 to 128, while fixing the number of PMs to 1024 (a fat-tree topology with 16-port switches). For experiment two, the number of applications is fixed to 16, whereas the number of PMs are varied from 16, 128, 1024 to 8192. In both of these experiments, we consider the number of critical and normal applications equal to each other. However, for experiment three, we vary the ratio of critical applications to all applications from 0% to 100% and set the number of applications and PMs to 16 and 1024, respectively. In all experiments, the number of VMs per application is set to $U[1,16]$ and the amount of traffic connection between mutual VMs is set to $U[1-100]$ Mbps, where $U[a,b]$ is a random uniform distribution between $a$ and $b$.

For PMs and VMs, we use real data. To this end, the characteristics of PMs are collected from different venders reported in SPECpowe_ssj2008 - third and fourth quarters of 2019 [43] (see 6) and the VMs' characteristics are get from the Microsoft Azure B-series [44] (see 7) and the Amazon EC2 A1-series [45] (see 8).

Table 6: PMs' Characteristics [43]

| **Vendor** | **Model** | **CPU [cores]** | **Memory [GB]** | $P^{full}$ **[W]** | $P^{idle}$ **[W]** |
|---|---|---|---|---|---|
| Inspur Corp. | NF5280M5 | 56 | 192 | 347 | 48.3 |
| Dell | R7515 | 64 | 128 | 246 | 99.5 |
| Hewlett Packard | DL385 Gen10 | 128 | 256 | 422 | 111 |
| Lenovo | SR850 | 112 | 384 | 672 | 121 |
| Fujitsu | LX1430 M1 | 64 | 256 | 250 | 60.8 |
| Supermicro Inc. | 1123US-TR4 | 56 | 192 | 385 | 48.6 |
| ASUSTeK Computer Inc. | RS720-E9-RS8 | 128 | 512 | 412 | 99.2 |

Table 7: Microsoft Azure B-series [44].

| **Name** | **vCPU [cores]** | **Memory [GB]** |
|---|---|---|
| **Standard˙B1s** | 1 | 1 |
| **Standard˙B1ms** | 1 | 2 |
| **Standard˙B2s** | 2 | 4 |
| **Standard˙B2ms** | 2 | 8 |
| **Standard˙B4ms** | 4 | 16 |
| **Standard˙B8ms** | 8 | 32 |
| **Standard˙B12ms** | 12 | 48 |
| **Standard˙B16ms** | 16 | 64 |
| **Standard˙B20ms** | 20 | 80 |

Table 8: Amazon EC2 A1-series [45].

| **Name** | **vCPU [cores]** | **Memory [GB]** |
|---|---|---|
| **a1.medium** | 1 | 2 |
| **a1.large** | 2 | 4 |
| **a1.xlarge** | 4 | 8 |
| **a1.2xlarge** | 8 | 16 |
| **a1.4xlarge** | 16 | 32 |

### 6.2. Benchmark algorithms

To demonstrate the effectiveness of the proposed method (JointPT+JointPR), we compare it to the following three state-of-the-art algorithms. First Fit Decreasing (FFD) [31], Energy-efficient and QoS-aware Virtual Machine Placement (EQVMP) [22], and Priority-aware VM Allocation (PAVA) [4]. In the FFD algorithm, VMs of a requested application is initially sorted in decreasing order of their CPU requirements. Then for each VM, it searches the list of PMs to host the VM based on the first fit manner, i.e., the VM is hosted on the first PM, which has enough resource capacity. The description of EQVMP and PAVA is presented in Subsection 2.2. Table 9 compares the time complexity of the algorithms. It is worth mentioning that PAVA and our proposed method are priority-aware while the other two algorithms do not consider the priority of applications.

Table 9: Time complexity comparison. $n$:= number of VMs of a requested application. $r$:= number of switch ports. $N$:= number of PMs. $E^{max}$:= size of $E$.

| | **FFD** | **EQVMP** | **PAVA** | **Proposed** |
|---|---|---|---|---|
| Critical Application | $O(n \times logn + n \times N)$ | $O(n^4 + n \times logn + n \times N)$ | $O(n \times N \times logN)$ | $O(N \times \log(r/2) + E^{max} \times \log n + n \times N)$ |
| Normal Application | $O(n \times logn + n \times N)$ | $O(n^4 + n \times logn + n \times N)$ | $O(n \times N^2 \times logN)$ | $O(N \times \log(r/2) + n \times N)$ |

### 6.3. Results

In this part, we present the results of the paper.

#### 6.3.1. Impact of varying the number of applications

Table 10 shows the network consumption versus the number of applications and reports the amount of consumed network resources by critical and normal applications alongside with their summation. As we can see from the results, the proposed algorithm and PAVA gives a better network consumption for critical applications. This is expected since our algorithm takes into account the traffic demand between VMs of critical applications and allocate them into PMs with the high computing resources and the closest proximity. Although PAVA ignores the traffic demand among VMs, it places VMs of critical applications onto the Rack which provides more computing resources. EQVMP performs well in terms of network consumption for normal applications. This is because of the hop reduction phase of this algorithm which applies it for both critical and normal applications Where neither the proposed algorithm nor PAVA uses this strategy for normal applications. But, compared to PAVA, our algorithm offers far less network consumption for normal applications. This is because of the sorting phase of the PAVA algorithm for normal application which sorts hosts in ascending order by their available resources. This leads to dispersing the VMs of a normal application across many PMs. Interestingly, the proposed algorithm outperforms the others in terms of total network consumption.

Table 10: Network consumption vs. number of applications. C:= network consumption of critical applications. N:= network consumption of normal application. T:=total network consumption.

| | **FFD** | | | **EQVMP** | | | **PAVA** | | | **Proposed** | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **#Applications** | $\mathbb{C}$ | $\mathbb{N}$ | $\mathbb{T}$ | $\mathbb{C}$ | $\mathbb{N}$ | $\mathbb{T}$ | $\mathbb{C}$ | $\mathbb{N}$ | $\mathbb{T}$ | $\mathbb{C}$ | $\mathbb{N}$ | $\mathbb{T}$ |
| **4** | 1,492 | 1,332 | 2,824 | 1,279 | **1,245** | 2,525 | 959 | 3,526 | 4,485 | **476** | 1,254 | **1,730** |
| **8** | 3,171 | 3,138 | 6,309 | 2,746 | **3,060** | 5,806 | 2,061 | 6,314 | 8,374 | **1,719** | 3,670 | **5,389** |
| **16** | 6,728 | 7,308 | 14,035 | 6,091 | **6,376** | 12,467 | 3,767 | 18,256 | 22,023 | **3,667** | 8,040 | **11,707** |
| **32** | 13,784 | 12,686 | 26,470 | 12,889 | **10,879** | 23,768 | **8,528** | 40,263 | 48,790 | 8,836 | 14,104 | **22,940** |
| **64** | 28,888 | 26,387 | 55,275 | 27,203 | **24564** | 51,767 | **16,176** | 80,672 | 96,848 | 18,346 | 31,135 | **49,481** |
| **128** | 56,095 | 52,111 | 108,206 | 52,271 | **49,432** | 101,703 | **29,836** | 148,621 | 178,457 | 31,056 | 67,084 | **98,140** |

Fig. 5 and Fig. 6 respectively illustrates the power consumption and resource wastage of the considered data center. Compared to the benchmarks, the proposed algorithm demonstrates better performance in both aspects. The main reason behind the reduction in the consumption of power is that our algorithm prioritizing the most power efficient Pods, Racks and PMs inside each Rack for VM placement. Furthermore, it hosts the normal application VMs in a way that the wastage of resources is minimized which reduces the number of activated PMs. It is worth mentioning that since PAVA attempts to allocate each set of critical application VMs into the host groups with high computing resources, so a lot of resources is wasted.

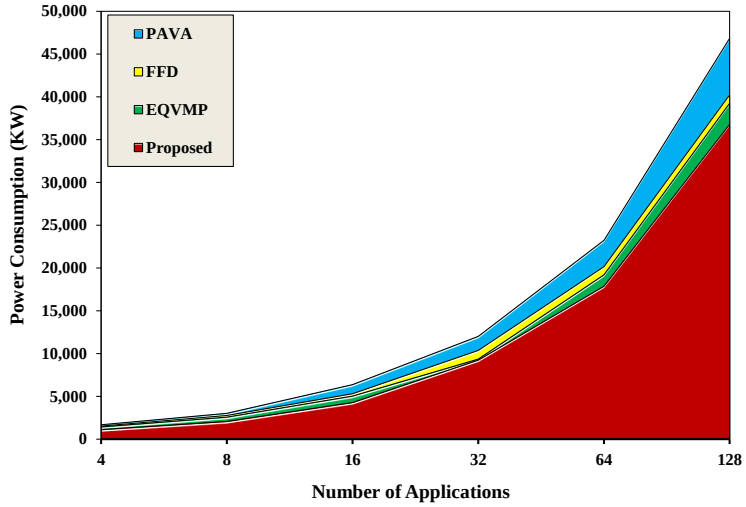


Figure 5: Impact of varying number of applications on power consumption

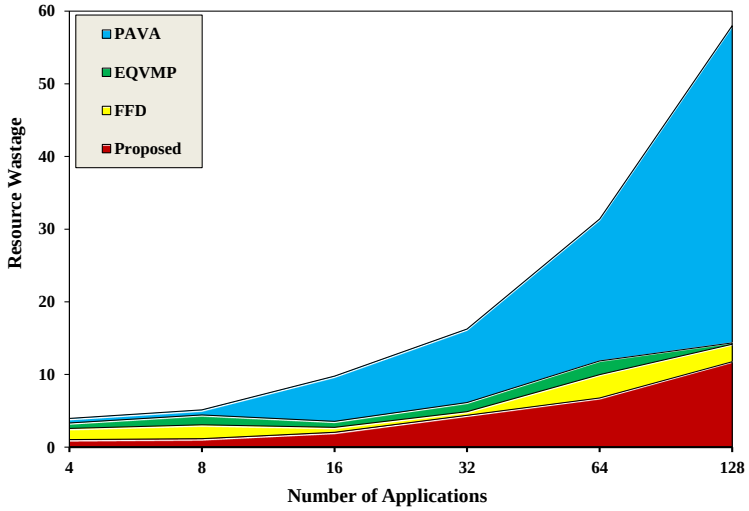


Figure 6: Impact of varying number of applications on resource wastage

*6.3.2. Impact of varying the number of PMs*

Table 11, Fig. 7 and Fig. 8 report the results of this experiment, where they respectively demonstrate the network consumption, power consumption and resource wastage for different topology scales with the same number of applications, i.e., 16. From Table 11 it can be inferred that PAVA has the lowest network consumption for critical applications, as its goal is to host VMs of critical applications on the host group which provides more computing resources. EQVMP gives the best results for normal ones, which is because of the hop reduction phase of this algorithm. However, the proposed algorithm achieves the lowest total network consumption in all cases. The reason is that our algorithm gives priority to PMs whose offer high computing resources and low power consumption. Therefore, more VMs can be host on a single PM, where this also leads to less power consumption (see Fig. 7). Additionally, our algorithm uses a traffic-aware approach for critical applications which resulting in significantly reducing the consumed network. Due to resource wastage-awareness for normal applications, the proposed algorithm outperforms the benchmark in this aspect (see Fig. 8).

Table 11: Network consumption vs. number of PMs. $C$:= network consumption of critical applications. $N$:= network consumption of normal application. $T$:=total network consumption.

| | **FFD** | | | **EQVMP** | | | **PAVA** | | | **Proposed** | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **#PMs** | $\mathbb{C}$ | $\mathbb{N}$ | $\mathbb{T}$ | $\mathbb{C}$ | $\mathbb{N}$ | $\mathbb{T}$ | $\mathbb{C}$ | $\mathbb{N}$ | $\mathbb{T}$ | $\mathbb{C}$ | $\mathbb{N}$ | $\mathbb{T}$ |
| **16** | 10,240 | 8,851 | 19,091 | 9,460 | **7,758** | 17,218 | **5,381** | 19,248 | 24,629 | 6,652 | 9,624 | **16,276** |
| **128** | 6,706 | 7,143 | 13,849 | 6,381 | **6,641** | 13,022 | **3,600** | 16,036 | 19,636 | 4,151 | 7,796 | **11,947** |
| **1024** | 6,728 | 7,308 | 14,035 | 6,091 | **6,376** | 12,467 | 3,767 | 18,256 | 22,023 | **3,667** | 8,040 | **11,707** |
| **8192** | 5,522 | 6,652 | 12,175 | 4,876 | **6,236** | 11,112 | **2,462** | 16,728 | 19,190 | 3,214 | 7,254 | **9,496** |

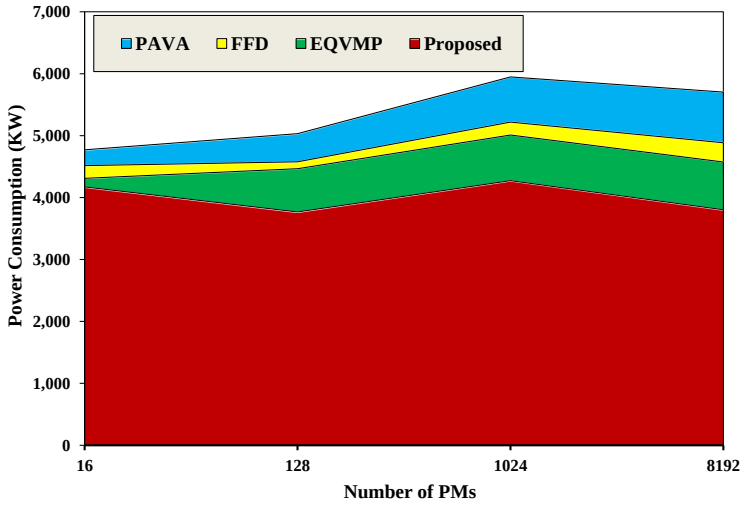


Figure 7: Impact of varying number of applications on resource wastage

*6.3.3. Impact of varying the ratio of critical applications*

This experiment investigates the performance of the different algorithms for various ratio of critical applications. Table 12 presents the results obtained for the network consumption of the considered data center. Generally speaking, for the low ratio, the proposed algorithm outperforms the comparison algorithms in terms of the consumed of network resources by critical applications. The reason behind this is that when the number of critical applications is low, their VMs get more chance to host on new PMs; because the most of active PMs are used by the large number of normal applications VMs. However, as this ratio is increases, PAVA provides the best results for network consumption of critical applications. As we stated in the previous experiments, PAVA tries to host VMs of each critical application on the host group with the highest available computing resources which leads to low network consumption for critical

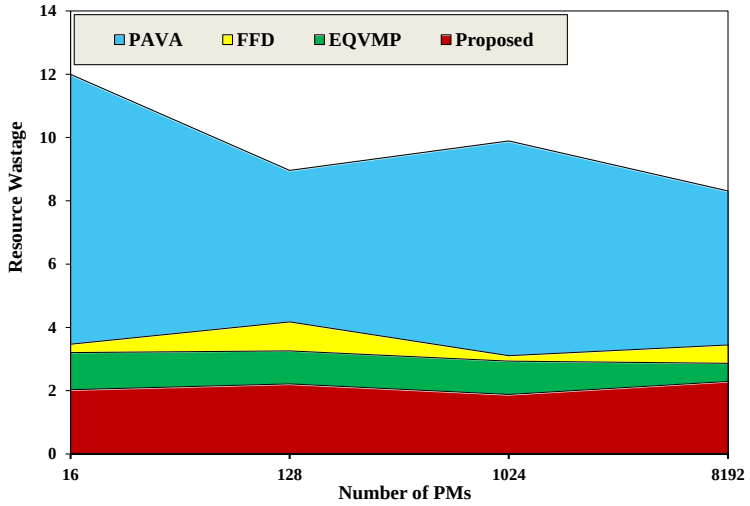


Figure 8: Impact of varying number of applications on resource wastage

applications. In all cases, EQVMP has the lowest value for the network consumption caused by normal applications as this algorithm consider the same strategy for both critical and normal applications. However, again our algorithm obtains the first rank in the total network consumption aspect.

Table 12: Network consumption vs. ratio of critical applications. $C$:= network consumption of critical applications. $N$:= network consumption of normal application. $T$:=total network consumption.

| | **FFD** | | | **EQVMP** | | | **PAVA** | | | **Proposed** | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $\mathbb{C}$ | $\mathbb{N}$ | $\mathbb{T}$ | $\mathbb{C}$ | $\mathbb{N}$ | $\mathbb{T}$ | $\mathbb{C}$ | $\mathbb{N}$ | $\mathbb{T}$ | $\mathbb{C}$ | $\mathbb{N}$ | $\mathbb{T}$ |
| **0%** | 0 | 15,507 | 15,507 | 0 | **14,185** | **14,185** | 0 | 38,114 | 38114 | 0 | 16,850 | 16,850 |
| **20%** | 2,936 | 11,496 | 14,432 | 2,502 | **10,137** | **12,639** | 1,966 | 26,789 | 28,756 | **1,190** | 12,843 | 14,034 |
| **40%** | 4,392 | 7,761 | 12,154 | 4,159 | **7,068** | **11,227** | 2,549 | 21,949 | 24,498 | **2,501** | 10,208 | 12,709 |
| **60%** | 7,646 | 4,761 | 12,407 | 6,831 | **4,305** | 11,136 | **4,826** | 15,678 | 20,503 | 5,105 | 5,963 | **11,068** |
| **80%** | 10,218 | 3,673 | 13,891 | 9,335 | **2,941** | 12,276 | **5,437** | 10,665 | 16,102 | 6,955 | 4,153 | **11,108** |
| **100%** | 13,021 | 0 | 13,021 | 11,098 | 0 | 11,098 | **6,157** | 0 | **6,157** | 8,503 | 0 | 8,503 |

By the analysis of Fig. 9 and Fig. 10, we conclude that the proposed algorithm gives the best results in terms of the consumption of power and wastage of resources, respectively. As previously stated, this is because of prioritizing the most power efficient PMs and considering the resource wastage in our objective function. Here we should note an important point. As it can be observed from Fig. 9, by increasing the ratio of critical applications, the power consumption of PAVA is decreased. This can be justified using the fact that PAVA places VMs of critical applications on the most powerful PMs; therefore, it uses the fewer PMs compared to the others, which leads to reducing power consumption. However, this policy dramatically increases the resource wastage (see Fig. 10). The reason behind this is as follows. For each critical application, PAVA maps their VMs on the Rack with more computing resources. After placing that VMs on the selected Rack, the available computing resources of the Rack is reduced and more probably it may not be selected for other critical applications. However, most likely, there are some PMs inside that Rack which are activated but their resources are not well used.

## 7. Discussion

In this work, the questions arised in the introduction are answered as follows. First, we proposed JointPT for critical (higher-priority) applications which tries to place mutual VMs with the higher traffic demand on PMs with the closest proximity. Therefore, critical applications enjoy low network latency, and thus high QoS. Second, to minimize the power consumption of a CDC, we prioritize the power-efficient PMs and reduce the number of active PMs. Also,

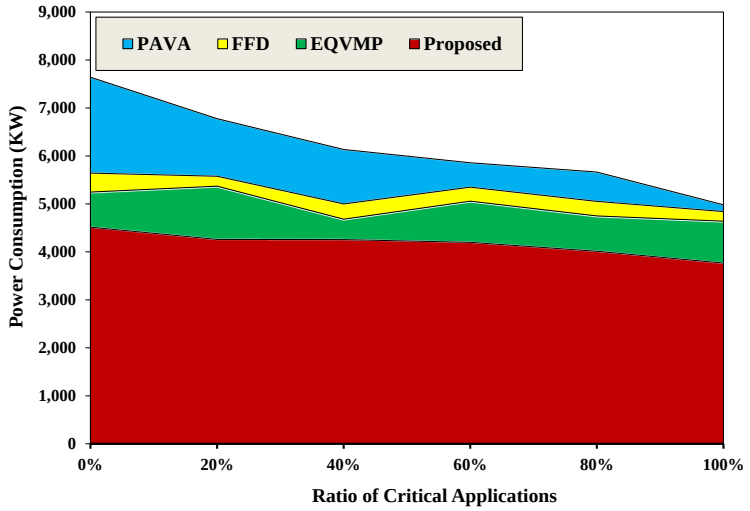


Figure 9: impact of varying number of applications on resource wastage

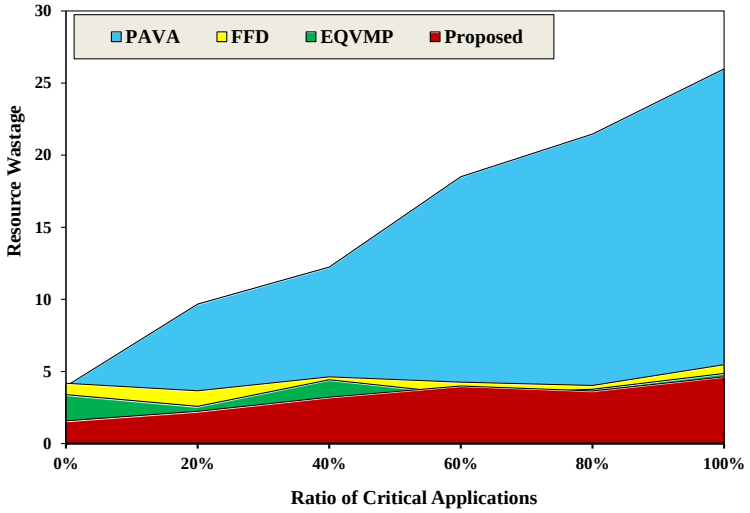


Figure 10: Impact of varying number of applications on resource wastage

to reduce the resource wastage, we proposed JointPR whose main objective is to host VMs of normal applications on the PMs with the least resource wastage. Finally, our proposed algorithms have low running time which make them suitable even for use in large-scale CDCs.

Although the present work provides an efficient solution for VMP problem, there are still several aspects concerning this problem. First, the proposed method can be extended to include the allocation of network bandwidth to provide high QoS for critical applications and avoid traffic congestion for normal ones. Second, in this work, we assume uniform random traffic among VMs of a requested application. However, to improve it, we can incorporate learning mechanisms for traffic prediction among VMs of requested applications in our future work. Third, this work is focused on the VMP problem. However, to improve the energy efficiency of a DCN more and more, an efficient VM consolidation phase also can be added to this work. Fourth, in this paper, we focused on the fat-tree topology. However, with a little modification, it can be extended to cope with the generalized network topologies. Finally, the proposed method cannot be directly applied in Fog/Edge-Cloud computing environments [46, 47]. To cope with such environments, we are planing to extend our proposed method and compare it with different learning methods such as those presented in [48, 49].

## 8. Conclusions and Future Work

In this paper, we studied the problem of VM placement of IoT applications in cloud data centers. We first formulated the problem as an mixed integer linear programming model with the goals of minimizing power consumption, network consumption and resource wastage. Then, we proposed a priority-aware scheme to efficiently solve the problem. To validate the effectiveness of our approach, we conducted extensive experiments and verify that the proposed approach shows better performance compared to the state-of-the-art. As future work, we aim to propose a meta-heuristic algorithm hybridized with a new bandwidth allocation mechanism to tackle this problem. Another interesting research direction is to extend the proposed method in a way that it can be applied in the integrated IoT-Fog-Cloud infrastructure [50, 51]. To this end, our system model should be improved to capture more prioritization attributes for applications and experiments can be conducted by the well-known frameworks like FogBus [52]. Moreover, as another research direction, we plan to modify our proposed heuristic algorithms to be applied for the allocation of micro-service applications in container-based clouds [49, 53].

## References


[1] C. P. Chen, C.-Y. Zhang, Data-intensive applications, challenges, techniques and technologies: A survey on big data, Information sciences 275 (2014) 314–347.

[2] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, M. Ayyash, Internet of things: A survey on enabling technologies, protocols, and applications, IEEE communications surveys & tutorials 17 (4) (2015) 2347–2376.

[3] K. Hwang, Cloud computing for machine learning and cognitive applications, MIT Press, 2017.

[4] J. Son, R. Buyya, Priority-aware vm allocation and network bandwidth provisioning in software-defined networking (sdn)-enabled clouds, IEEE Transactions on Sustainable Computing 4 (1) (2018) 17–28.

[5] M. R. Jabbarpour, A. Marefat, A. Jalooli, H. Zarrabi, Could-based vehicular networks: a taxonomy, survey, and conceptual hybrid architecture, Wireless Networks 25 (1) (2019) 335–354.

[6] Z. Li, J. Ge, H. Hu, W. Song, H. Hu, B. Luo, Cost and energy aware scheduling algorithm for scientific workflows with deadline constraint in clouds, IEEE Transactions on Services Computing 11 (4) (2015) 713–726.

[7] P. Vadakkepat, P. Lim, L. C. De Silva, L. Jing, L. L. Ling, Multimodal approach to human-face detection and tracking, IEEE transactions on industrial electronics 55 (3) (2008) 1385–1393.

[8] H. Talebian, A. Gani, M. Sookhak, A. A. Abdelatif, A. Yousafzai, A. V. Vasilakos, F. R. Yu, Optimizing virtual machine placement in iaas data centers: taxonomy, review and open issues, Cluster Computing (2019) 1–42.

[9] W. Lin, W. Wu, 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 Transactions on Services Computing (2019).

[10] M. C. Silva Filho, C. C. Monteiro, P. R. Inácio, M. M. Freire, Approaches for optimizing virtual machine placement and migration in cloud environments: A survey, Journal of Parallel and Distributed Computing 111 (2018) 222–250.

[11] N. Donyagard Vahed, M. Ghobaei-Arani, A. Souri, Multiobjective virtual machine placement mechanisms using nature-inspired metaheuristic algorithms in cloud environments: A comprehensive review, International Journal of Communication Systems 32 (14) (2019) e4068.

[12] M. Masdari, S. Gharehpasha, M. Ghobaei-Arani, V. Ghasemi, Bio-inspired virtual machine placement schemes in cloud computing environment: taxonomy, review, and future research directions, Cluster Computing (2019) 1–31.

[13] A. Beloglazov, J. Abawajy, R. Buyya, Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing, Future generation computer systems 28 (5) (2012) 755–768.

[14] M. Tang, S. Pan, A hybrid genetic algorithm for the energy-efficient virtual machine placement problem in data centers, Neural Processing Letters 41 (2) (2015) 211–221.
[15] M. Mohammadhosseini, A. T. Haghighat, E. Mahdipour, An efficient energy-aware method for virtual machine placement in cloud data centers using the cultural algorithm, The Journal of Supercomputing 75 (10) (2019) 6904–6933.
[16] A. Jayanetti, R. Buyya, J-opt: A joint host and network optimization algorithm for energy-efficient workflow scheduling in cloud data centers, in: Proceedings of the 12th IEEE/ACM International Conference on Utility and Cloud Computing, 2019, pp. 199–208.
[17] S. Mishra, S. Jain, Ontologies as a semantic model in iot, International Journal of Computers and Applications 42 (3) (2020) 233–243.
[18] Y. Gao, H. Guan, Z. Qi, Y. Hou, L. Liu, A multi-objective ant colony system algorithm for virtual machine placement in cloud computing, Journal of Computer and System Sciences 79 (8) (2013) 1230–1242.
[19] Q. Zheng, R. Li, X. Li, N. Shah, J. Zhang, F. Tian, K.-M. Chao, J. Li, Virtual machine consolidated placement based on multi-objective biogeography-based optimization, Future Generation Computer Systems 54 (2016) 95–122.
[20] S. Azizi, M. Shojafar, J. Abawajy, R. Buyya, Grvmp: A greedy randomized algorithm for virtual machine placement in cloud data centers, IEEE Systems Journal (2020).
[21] X. Meng, V. Pappas, L. Zhang, Improving the scalability of data center networks with traffic-aware virtual machine placement, in: 2010 Proceedings IEEE INFOCOM, IEEE, 2010, pp. 1–9.
[22] S.-H. Wang, P. P.-W. Huang, C. H.-P. Wen, L.-C. Wang, Eqvmp: Energy-efficient and qos-aware virtual machine placement for software defined datacenter networks, in: The International Conference on Information Networking 2014 (ICOIN2014), IEEE, 2014, pp. 220–225.
[23] S. Farzai, M. H. Shirvani, M. Rabbani, Multi-objective communication-aware optimization for virtual machine placement in cloud datacenters, Sustainable Computing: Informatics and Systems (2020) 100374.
[24] M.-H. Malekloo, N. Kara, M. El Barachi, An energy efficient and sla compliant approach for resource allocation and consolidation in cloud computing environments, Sustainable Computing: Informatics and Systems 17 (2018) 9–24.
[25] M. Cheng, J. Li, S. Nazarian, Drl-cloud: Deep reinforcement learning-based resource provisioning and task scheduling for cloud service providers, in: 2018 23rd Asia and South Pacific Design Automation Conference (ASP-DAC), IEEE, 2018, pp. 129–134.
[26] A. Ghasemi, A. T. Haghighat, A multi-objective load balancing algorithm for virtual machine placement in cloud data centers based on machine learning, COMPUTING (2020).
[27] M. Tarahomi, M. Izadi, M. Ghobaei-Arani, An efficient power-aware vm allocation mechanism in cloud data centers: a micro genetic-based approach, Cluster Computing (2020) 1–16.
[28] H. Zhao, Q. Wang, J. Wang, B. Wan, S. Li, Vm performance maximization and pm load balancing virtual machine placement in cloud, in: 2020 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGRID), IEEE, 2020, pp. 857–864.
[29] S. Tuli, S. S. Gill, G. Casale, N. R. Jennings, ithermofog: Iot-fog based automatic thermal profile creation for cloud data centers using artificial intelligence techniques, Internet Technology Letters (2020) e198.
[30] M. Mitzenmacher, E. Upfal, Probability and computing: Randomization and probabilistic techniques in algorithms and data analysis, Cambridge university press, 2017.
[31] G. Keller, M. Tighe, H. Lutfiyya, M. Bauer, An analysis of first fit heuristics for the virtual machine relocation problem, in: 2012 8th international conference on network and service management (cnsm) and 2012 workshop on systems virtualiztion management (svm), IEEE, 2012, pp. 406–413.
[32] D. S. Dias, L. H. M. Costa, Online traffic-aware virtual machine placement in data center networks, in: 2012 Global Information Infrastructure and Networking Symposium (GIIS), IEEE, 2012, pp. 1–8.
[33] C. Gao, H. Wang, L. Zhai, Y. Gao, S. Yi, An energy-aware ant colony algorithm for network-aware virtual machine placement in cloud computing, in: 2016 IEEE 22nd International Conference on Parallel and Distributed Systems (ICPADS), IEEE, 2016, pp. 669–676.
[34] R. A. da Silva, N. L. da Fonseca, Topology-aware virtual machine placement in data centers, Journal of Grid Computing 14 (1) (2016) 75–90.
[35] X. Li, Z. Lian, X. Qin, W. Jie, Topology-aware resource allocation for iot services in clouds, IEEE Access 6 (2018) 77880–77889.
[36] G. Cao, Topology-aware multi-objective virtual machine dynamic consolidation for cloud datacenter, Sustainable Computing: Informatics and Systems 21 (2019) 179–188.
[37] M. Al-Fares, A. Loukissas, A. Vahdat, A scalable, commodity data center network architecture, ACM SIGCOMM computer communication review 38 (4) (2008) 63–74.
[38] D. Belabed, S. Secci, G. Pujolle, D. Medhi, Striking a balance between traffic engineering and energy efficiency in virtual machine placement, IEEE Transactions on Network and Service Management 12 (2) (2015) 202–216.
[39] S. Azizi, M. Zandsalimi, D. Li, An energy-efficient algorithm for virtual machine placement optimization in cloud data centers, Cluster Computing 23 (4) (2020) 3421—-3434.
[40] S. Pelley, D. Meisner, T. F. Wenisch, J. W. VanGilder, Understanding and abstracting total data center power, in: Workshop on Energy-Efficient Design, Vol. 11, 2009, pp. 1–6.
[41] Z. Á. Mann, Approximability of virtual machine allocation: much harder than bin packing, in: 9th Hungarian-Japanese Symposium on Discrete Mathematics and Its Applications, 2015, pp. 21–30.
[42] S. Omer, S. Azizi, M. Shojafar, R. Tafazolli, Priority, power, and traffic-aware vmp algorithm source code, `https://github.com/mshojafar/sourcecodes/tree/master/Shvan2020VMPJSA_Sourcecode` (2020).
[43] SPECpower˙ssj2008 Results, `https://www.ibm.com/analytics/data-science/prescriptive-analytics/cplex-optimizer` (Accessed: 2020-11-03).
[44] Microsoft Azure. General purpose virtual machine sizes., `https://docs.microsoft.com/en-us/azure/virtual-machines/windows/sizes` (Accessed: 2020-11-03).
[45] Amazon EC2. Amazon ec2 instance types., `https://aws.amazon.com/ec2/instance-types/?nc1=h_ls` (Accessed: 2020-11-03).
[46] C.-H. Hong, B. Varghese, Resource management in fog/edge computing: a survey on architectures, infrastructure, and algorithms, ACM Computing Surveys (CSUR) 52 (5) (2019) 1–37.
[47] B. Omoniwa, R. Hussain, M. A. Javed, S. H. Bouk, S. A. Malik, Fog/edge computing-based iot (feciot): Architecture, applications, and research issues, IEEE Internet of Things Journal 6 (3) (2018) 4118–4149.

[48] S. Tuli, S. Ilager, K. Ramamohanarao, R. Buyya, Dynamic scheduling for stochastic edge-cloud computing environments using a3c learning and residual recurrent neural networks, IEEE Transactions on Mobile Computing (2020).

[49] L. Espeholt, H. Soyer, R. Munos, K. Simonyan, V. Mnih, T. Ward, Y. Doron, V. Firoiu, T. Harley, I. Dunning, et al., Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures, in: International Conference on Machine Learning, 2018, pp. 1407—-1416.

[50] A. Yousefpour, C. Fung, T. Nguyen, K. Kadiyala, F. Jalali, A. Niakanlahiji, J. Kong, J. P. Jue, All one needs to know about fog computing and related edge computing paradigms: A complete survey, Journal of Systems Architecture 98 (2019) 289–330.

[51] M. Abbasi, E. M. Pasand, M. R. Khosravi, Workload allocation in iot-fog-cloud architecture using a multi-objective genetic algorithm, Journal of Grid Computing (2020) 1–14.

[52] S. Tuli, R. Mahmud, S. Tuli, R. Buyya, Fogbus: A blockchain-based lightweight framework for edge and fog computing, Journal of Systems and Software 154 (2019) 22–36.

[53] B. Tan, H. Ma, Y. Mei, A nsga-ii-based approach for multi-objective micro-service allocation in container-based clouds, in: 2020 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGRID), IEEE, 2020, pp. 282–289.

**Biographies**

**Shvan Omer** received his B.Sc. degree of Computer science and M.Sc. degree of computer engineering (Artificial Intelligence) at the University of Kurdistan, Sanandaj, Iran, in 2017 and 2020, respectively. His main research interests include Internet of Things (IoT), Green Computing, and Cloud Data Centers. Currently, he is a member of Internet of Things and Computing Systems Research Laboratory (IoTComSys Lab) at the University of Kurdistan.

**Sadoon Azizi** 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 and IT, University of Kurdistan, Sanandaj, Iran. He is also the director of the Internet of Things and Computing Systems Research Laboratory (IoTComSys Lab) and Head of the High-Performance Computing (HPC) center at the University of Kurdistan. His current research interests include Cloud Computing, Fog/Edge Computing, Internet of Things, and Heuristic Algorithms for Combinatorial Optimization Problems. For additional information: `https://research.uok.ac.ir/~sazizi/en/`

**Mohammad Shojafar** is a Senior Lecturer (Associate professor) in the Network Security and an Intel Innovator working in the 6G Innovation Centre (6GIC) at the University of Surrey, UK. Before joining 6GIC, he was a Senior Researcher and a Marie Curie Fellow at the University of Padua, Italy. Dr. Mohammad was a PI of PRISENODE project, a 275,000 euro Horizon 2020 Marie Curie project in the areas of Fog/Cloud security collaborating at the University of Padua. He also was a PI on an Italian SDN security and privacy (60,000 euro) supported by the University of Padua in 2018 and a Co-PI on an Ecuadorian-British project on IoT and Industry 4.0 resource allocation (20,000 dollars) in 2020. He was contributed to some Italian projects in telecommunications like GAUChO, SAMMClouds, and SC2. In 2016, he received his PhD in ICT from the Sapienza University of Rome, Italy, with an assessment of 'Excellent'. He is an Associate Editor in IEEE Transactions on Consumer Electronics and IET Communications. For additional information: `http://mshojafar.com`

**Rahim Tafazolli** is the Regius Professor, FREng of mobile and satellite communications, since April 2000. He has also been the Director of ICS, since January 2010, and the Founder and the Director with the 6G Innovation Centre (6GIC), University of Surrey, United Kingdom. He has more than 25 years of experience in digital communications research and teaching. He has authored and co-authored more than 500 research publications. He is a co-inventor on more than 30 granted patents, all in the field of digital communications. He is regularly invited to deliver keynote talks and distinguished lectures to International conferences and workshops. In 2011, he was appointed as a Fellow of Wireless World Research Forum (WWRF) in recognition of his personal contributions to the wireless world and the heading one of Europe leading research groups. Many governments regularly invite him for advice on 5G technologies. He was an Advisor to the Mayor of London in regard to the London Infrastructure Investment 2050 Plan, from May to June 2014. He has given many interviews to international media in the form of television, radio interviews, and articles in the international press. For additional information: `https://www.surrey.ac.uk/people/rahim-tafazolli`