Cluster Computing (2020) 23:837–878
https://doi.org/10.1007/s10586-019-02954-w

Optimizing virtual machine placement in IaaS data centers: taxonomy, review and open issues

Hamid Talebian[1] · Abdullah Gani[1] · Mehdi Sookhak[5] · Ahmed Abdelaziz Abdelatif[3] · Abdullah Yousafzai[6] · Athanasios V. Vasilakos[4] · Fei Richard Yu[2]

Received: 7 August 2015 / Revised: 19 May 2019 / Accepted: 20 June 2019 / Published online: 27 July 2019

Abstract
The unprecedented growth of energy consumption in data centers created critical concern in recent years for both the research community and industry. Besides its direct associated cost; high energy consumption also results in a large amount of CO_2 emission and incurs extra cooling expenditure. The foremost reason for overly energy consumption is the underutilization of data center resources. In modern data centers, virtualization provides a promising approach to improve the hardware utilization level. Virtual machine placement is a process of mapping a group of virtual machines (VMs) onto a set of physical machines (PMs) in a data center with the aim of maximizing resource utilization and minimizing the total power consumption by PMs. An optimal virtual machine placement algorithm substantially contributes to cutting down the power consumption through assigning the input VMs to a minimum number of PMs and allowing the dispensable PMs to be turned off. However, VM Placement Problem is a complex combinatorial optimization problem and known to be NP-Hard problem. This paper presents an extensive review of virtual machine placement problem along with an overview of different approaches for solving virtual machine placement problem. The aim of this paper is to illuminate challenges and issues for current virtual machine placement techniques. Furthermore, we present a taxonomy of virtual machine placement based on various aspects such as methodology, number of objectives, operation mode, problem objectives, resource demand type and number of clouds. The state-of-the-art VM Placement techniques are classified in single objectives and multi-objective groups and a number of prominent works are reviewed in each group. Eventually, some open issues and future trends are discussed which serve as a platform for future research work in this domain.

Keywords Cloud computing · Data center · Energy · Consolidation · Virtual machine placement

✉ Hamid Talebian
talebian@um.edu.my

Abdullah Gani
abdullah@um.edu.my

Mehdi Sookhak
m.sookhak@ieee.org

[1] Centre for Mobile Cloud Computing (C4MCC), University of Malaya, Kuala Lumpur, Malaysia

[2] Department of Systems and Computer Engineering, Carleton University, Ottawa, Canada

[3] The Future University, Khartoum, Sudan

[4] Department of Computer Science, Luleå University of Technology, Luleå, Sweden

[5] School of Informaion Technology, Illinois State University, Normal, USA

[6] HITEC University, Taxila, Pakistan

1 Introduction

Cloud computing is a modern direction for computing as opposed to conventional desktop computing. This model received significant attention from both industry and academia and being emerged as a publicly accepted style of computing [1]. The cloud computing model offers a number of remarkable advantages over desktop computing. These benefits are location-independent access to the services, rapid elasticity, measured service, minimal capital investment, and lower maintenance cost.

A cloud service is composed of a set of data centers around the globe. Each data center contains thousands of servers and resides in a different geographic point for performance enhancement and reliability purpose while connected to other data centers with high-speed

telecommunication link [2]. High operational cost such as electricity cost, cooling cost, and footprint cost is the leading challenge for today's modern cloud data center providers. Managing operational cost enables cloud providers to offer affordable and competitive services to their customers [3]. However, among different costs, the rampant electricity cost constitutes the largest fraction of total cost in a data center and needs to be properly controlled [4–7]. The steady increase of energy consumption has appeared as a real concern in operating a data center. One of the main reasons for high energy consumption is inefficient management of resources. Particularly, most of the servers suffer from severe underutilization which is a major cause for power overconsumption [8, 9].

In modern data centers, virtualization as a primitive technology provides a potential solution to enhance resource utilization while it guarantees performance and isolation for cloud applications [10]. Even though virtualization technology offers a potential platform to improve resource utilization, there are still some issues which obstruct the maximum utilization of hardware. One of these issues is the inefficient deployment of virtual machines in a virtualized data center [11]. Having a data center with an enormous number of PMs, the basic challenge is how to map each input VMs to a most suitable PM so that the total electricity consumption by all active PMs is minimized and for each individual PM maximum resource utilization is achieved. Optimal placement of virtual machines is one of the prevailing approaches for efficient use of hardware in a data center and regarded as the core of cloud computing. Nonetheless, VM Placement is a complex computational task and recognized as an NP-Hard combinatorial optimization problem [12]. Despite past research works, the problem still needs more attention and further research effort should be devoted to this domain. This paper delineates the characteristics and significance of the VM Placement and presents a comprehensive review of different approaches for solving VM Placement. These approaches are either intended to optimize a sole VM Placement objective function or presumed to be a multi-objective optimization approach. For each of the two categories, various research works are overviewed. In addition, VM Placement solutions are discussed based on other perspectives and properties. Finally, a number of open issues are listed which can serve as a platform for further research in this domain. It is worth mentioning, many of the research works discussed in this paper could also apply to the private data centers and they are not exclusively designed for a cloud computing environment.

While to the best of our knowledge there are few related survey articles on the domain of this article, what makes the present paper relatively distinct is its main concentration on the optimization perspective. Because the majority of existing surveys overlooked or less concentrated on the optimization aspect, we particularly decided to discuss the problem under the context of a single/multi-objective optimization framework in detail. Two common approaches as weighted-sum approach and Pareto-based approach for dealing with the VM placement having multiple objectives are clearly explained and existing works are discussed, classified and summarized in accordance with these two common approaches. Furthermore, special attention is paid to the heuristic methodologies proposed to deal with the problem in various existing works. The heuristic methods presented in the literature are described, analyzed and compared including their strength and weakness. Lastly, unlike the majority of the existing works which have been published a few years prior to this study, the present work covers the state-of-the-art research works conducted within the domain. In the following, a few most recent and closely related survey articles are briefly compared to our present work.

Usmani and Singh [13] presented a study of VM placement techniques used in a green cloud. The survey's main focus is merely improving energy efficiency. The survey does not cover a range of VM placement objectives such as network traffic, resource wastage, and response time which are discussed in our work. Also, unlike the wide range of optimization methods presented in the present paper, only a limited number of mostly deterministic algorithms were discussed.

In another related survey, Masdari et al. [14] provided a review of VM placement schemes for cloud computing. In contrast to our work, the authors did not address VM placement from the optimization point of view along with its theoretical foundation and specification in a single and multi-objective variation. Moreover, the survey does not present the advantage and disadvantage of each existing method as presented in our work. On the contrary, our work provides an inclusive and well-structured taxonomy of different properties of VM placement along with in detail and organized summarization of single objective, multi-objective methods.

A systematic review of VM placement was conducted by [15]. Our work significantly differs from this work in the sense that the work, in fact, is a systematic and concise review of the relevant literature in the domain. It is mainly focused on collecting, organizing, and quantitatively summarizing the literature in a systematic way rather than a detailed analysis, discussion and conclusion of various VM placement schemes.

Pietri and Sakellariou [16] also surveyed the body of literature related to mapping VMs onto PMs. The survey is mainly structured on the basis of four different factors as VM configuration, VM placement, optimization objectives, and application metrics and tools. The optimization

objectives are classified into three categories of resource utilization, monetary units, and energy consumption. However, the survey's concentration on the optimization aspect of the problem (including theoretical representation, different approaches to deal with the problem and detailed methodology) is minimal.

The rest of this paper is organized as follows: in the next section, a background of the problem presented, including a summary of the cloud computing paradigm, virtualization technology and significance of energy consumption concern in modern large scale data centers. Section 3 defines the VM placement problem along with its characteristics. Section 4 reviews the state-of-the-art VM placement strategies classified into a single objective and multi-objective VM placement. Section 5 discusses the VM placement problem from different perspectives on the basis of the taxonomy presented in Fig. 19. A number of open issues for future research are listed in Sect. 6. Lastly, Sect. 7 concludes this paper.

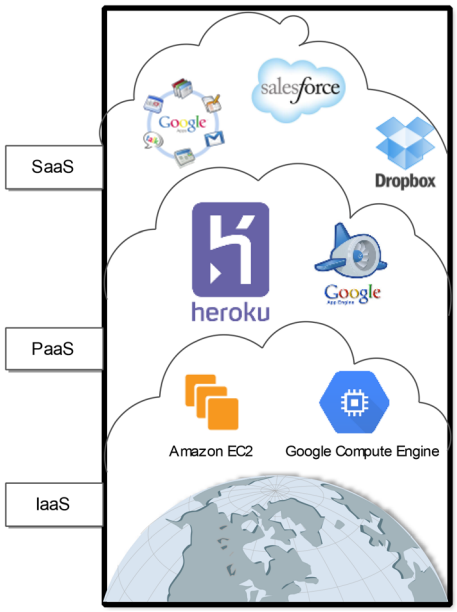

Fig. 1 Different service delivery models in cloud computing

2 Background

This section commences with an introduction to the concept of cloud computing as the modern paradigm of computing and particularly Infrastructure as a Service (IaaS) as underlying service. Then, an introductory to the virtualization as the fundamental technology of cloud computing is provided. Thereafter, the importance of escalating energy consumption in today's modern data centers is highlighted. Finally, in the last section, a general architecture for VM Placement in a data center is illustrated.

2.1 Cloud computing paradigm

In the cloud computing paradigm, rather than owning a local computing asset, users rely on a service they receive from distant high-performance provider [17]. A cloud facility is administrated by *Cloud Service Provider* (CSP) and consists of a large number of servers spread over different geographical points around the world. The cloud services are delivered via three different typical models. These models are Software as Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS).

Figure 1 shows different cloud service delivery models with examples for each mode. In the SaaS model, an application is hosted on the cloud server and users are given on-demand access to a ready application [18]. Examples of the SaaS are Google Apps, Dropbox, and Salesforce.com Applications. In PaaS, developers make use of an online programming framework to develop their application without the cost of the underlying software and hardware [18]. Heroku and Google Apps Engine are two instances for PaaS. In IaaS, computer equipment such as processor, memory, network, and storage are abstracted and delivered to the users as a service. Examples of IaaS are Amazon EC2 and Google Compute Engine. As IaaS environment serves as a platform for VM placement, in this paper we mainly focused on the IaaS cloud system.

IaaS emerges as one of the popular and powerful services in cloud computing [19]. An IaaS cloud provides the on-demand and scalable service to the users through a large shared pool of computing resources in the form of VMs while the users are charged based on a pay-as-you-go pricing model similar to water and electricity [20].

Due to the elasticity of IaaS services, the resources can be instantly and flexibly scaled up/down and therefore users are not required to anticipate their future hardware demands. This feature gives users the illusion of infinite computing resources without having to establish their own infrastructure [21]. The user specifies the hardware configuration of requested VM including processor speed, memory size, disk space amount and network bandwidth and the VMs are created accordingly, instantly and are run on the cloud provider's infrastructure [22]. From the customer's standpoint, these approach results in substantial cost effectiveness since the capital expenditure is eliminated. Moreover, improved reliability is yielded because the service is provided by the vigorous provider's infrastructure. With recent rapid development in IaaS market, this model has become a remarkable alternative to the local ownership of computing infrastructure and therefore a number of prestigious companies such as Netflix shifted to IaaS cloud instead of owning dedicated servers [23].

Examples of real-world IaaS providers are Amazon EC2 and GoGrid, Rackspace cloud. IaaS cloud system leverages the virtualization technology to manage the underlying computing resources and deliver flexible and dynamic service to the customers [24]. A schematic of an IaaS cloud is shown in Fig. 2.

2.2 Virtualization technology

The virtualization was first introduced in the late 1960s by IBM to make efficient use of expensive hardware in that time and mostly applied to the desktop sector [6, 25, 26]. A large underutilized mainframe's hardware was logically partitioned into slots which enabled users to use the resource in a time-sharing fashion [27]. Even though today's computer hardware is not as expensive as before, the virtualization technique is still being applied as a technique to divide the hardware resource of a single computer to multiple segregated computing environments [28]. Also, modern data centers make use of virtualization for other advantages [29]. These advantages are: reducing management complexity [29]. portability, encapsulation, isolation and efficient utilization through server consolidation [30]. Virtualization facilitates the way an administrator manages a data center. The isolation property prevents malicious applications, security flaws, and software failures to affect other co-located machines [29, 31]. Encapsulation ensures that the whole state of a machine as an image file can be cloned or migrated to another host with the purpose of load balancing, scheduling and fault tolerance in case of hardware failure [28, 31]. Virtualization also makes applications and services easily portable across heterogeneous hosts with different geographical locations. Again, virtualization provides finer-grained resource allocation [26]. However, the main application of virtualization in a data center is preventing server sprawl and efficient utilization of hardware through server consolidation [25]. Consolidation of many underutilized servers in a small number of host results in a significant saving of energy cost [28].

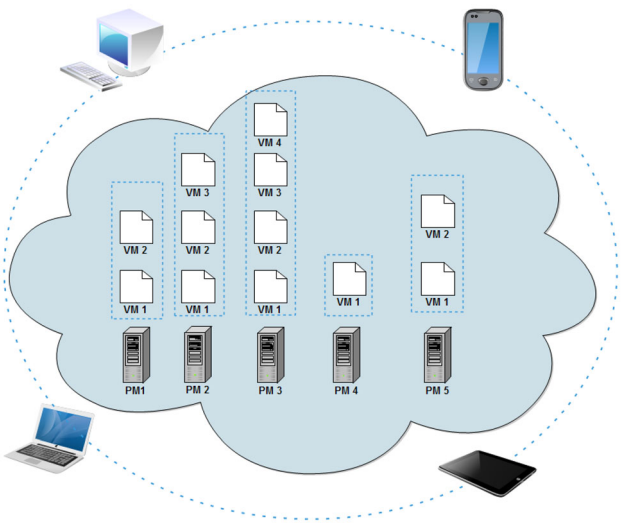

Fig. 2 IaaS cloud system model

In virtualization concept, on top of underlying hardware layer, a control program called *Hypervisor* or *Virtual Machine Monitor (VMM)* (such as VMware, KVM, Hyper-V, and Xen) creates, executes and manages virtualized instances of a machine. This virtual instance which is called a *virtual machine (VM)* contains its own operating system and applications and acts as the logical equivalent of a physical machine [32]. The applications that are installed on top of the virtualized machine are expected to perform identically as if they were installed on a real physical machine. The hypervisor tends to provide a transparent underlying hardware layer to the virtualized software and make application independent to a specific type of hardware and therefore resilient to the future hardware changes [6]. Multiple virtual machines can co-reside on a single physical machine and each virtual machine encapsulates a complete operating system bundled with the necessary applications.

2.3 Significance of energy consumption in modern data centers

The energy consumed in large scale data centers has dramatically risen during recent years. This alarming growth of energy consumption has caused deep concerns in industry and research communities. Kaplan et al. [33] denote each data center consumes electricity equal to 25,000 households on average. During 2008–2010 data centers in the United States consumed the energy provided by ten nuclear power stations [34]. It is reported that Google data centers consume power equal to the total consumption of a small city like San Francisco [35]. Again, Gartner estimates that energy cost for the IT industry will grow from 10 to 50% in the next few years [36]. Apart from directly associated electricity cost, over-consumption of electricity also requires larger expenditure for air conditioning computer systems since cooling cost is proportional to the energy spent for the computation purpose. Moreover, emission of carbon dioxide (CO_2) is the negative environmental effect of overly energy consumption and it is one of the causes of global warming and climate change. According to Gartner [37], 2% of CO_2 global emissions are from the IT industry. All these facts motivate researchers to move toward green data centers by minimizing the level of energy consumed in data centers.

By far, two approaches suggested for controlling the increasing level of electricity consumed in data centers.

The first approach mainly focuses on the design and manufacture of energy-efficient computer hardware such as processor, memory, and disk while the second approach relies on the efficient management of existing hardware systems. Indeed, the latter approach tends to make efficient use of the available hardware resources. One of the leading reasons for energy over-consumption in data centers is the underutilization of hardware resource. A study [6] shows that the utilization degree in a data center is between 10 to 50%. Underutilization often occurs as a result of over-provisioning which is an allocation of more than enough computing resources to the input workloads. Due to the uncertainty of future demands for an application, the resources are allocated based on the peak resource requirement of applications at the beginning. Energy consumption and low utilization are two correlated issues. Resources having low utilization still consume a non-trivial amount of power [38]. A larger number of underutilized servers in a data center results in higher energy consumption, more expense for cooling systems and extra required footprint. Therefore, reducing the number of poorly utilized machines to a minimum number of fully utilized ones contributes to cutting down the amount of electricity usage in a data center.

2.4 System architecture

The context for VM placement is a large scale data center with numerous physical servers inter-connected using high-speed telecommunication links. Each server is characterized by a certain amount of different resource types it has such as CPU, memory, disk storage, and bandwidth. Figure 3 shows the architecture of the system. The main roles in these systems are *cloud provide*r, *end-user* and *client* [39].

- *The cloud provider* is the owner of the infrastructure, manages the data center, its resources and lease them to the client.
- *The client* leases an infrastructure in the form of VMs for a certain period of time from a cloud provider.
- *End-user* uses the application ran on the cloud infrastructure.

It is to be mentioned that in case of multiple independent public cloud providers, an additional entity which is often called *Cloud Broker* provides intermediation service and allow users to deploy their VM across multiple providers [40]. However, the common interaction between the above roles is as follows [39]:

- A client plans to run his application on the cloud infrastructure and then submits his request for provisioning of a set of VMs with pre-determined hardware specification.
- Cloud provider, creates the requested collection of VMs accordingly.
- Having a determined set of VMs, the cloud provider decides how to assign each VM on a most suitable PM.
- Once the VMs are deployed on the different PMs on the data center they are available to serve requests from end users.

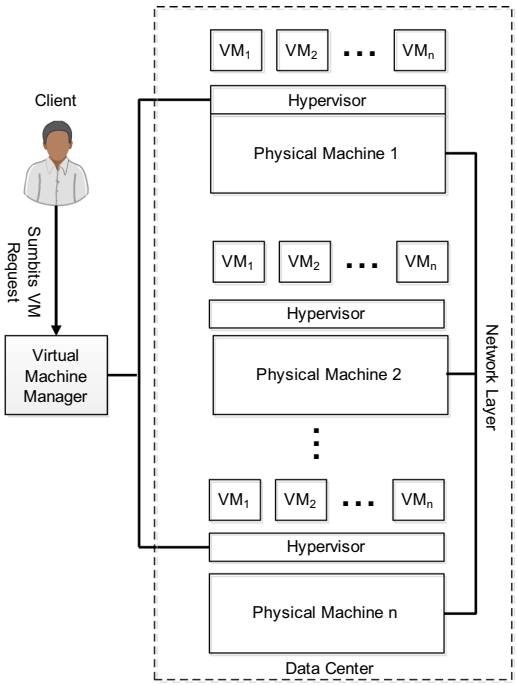

Fig. 3 System architecture

A software called Virtual Machine Manager (VMM) (or hypervisor) continuously monitors the available resources in each PM and based on its placement algorithm places a requested set of VMs on a certain subset of available PMs in a data center. The virtual machine manager is comprised of two different modules: the *local manager* and *global manager* [41]. The local manager operates on each individual PM. The role of a local manager is to monitor the current residual resource capacity in the PM together with resource utilization and reports the statistics to the global manager. The global manager module resides on the master PM and receives and compiles information coming from each local manager. Global manager process those reports to make a global picture of current resource usage in the whole data center. In addition, the global manager is responsible for optimizing VM placement.

3 Problem definition

The virtual machine placement is defined as follows: given a set $V = \{v_1, v_2, \ldots, v_m\}$ of virtual machines and a set $P = \{p_1, p_2, \ldots, p_n\}$ of physical machines, the goal is to find a specific mapping of VMs in V into PMs in P that most minimizes/maximizes a certain predefined objective(s). The most common objective is minimizing the number of running PMs. However, the other objectives such as network traffic also can be defined. Each VM demands a different amount from each resources type (i.e. CPU, memory, and disk space and network bandwidth). On the other hand, each PM has a certain capacity from each resource type. It is assumed that VMs do not demand more resource than a single PM can offer [42].

For a large data center with thousands of PMs, assigning VMs to PMs is an intricate decision-making task for a human administrator. This is due to the presence of enormous potential mappings while only one or few of these mapping results in an optimal value of the predetermined objective. Theoretically, VM Placement is known to be an NP-Hard combinatorial optimization problem [12, 43–45] since there is no provable efficient algorithm to solve it [46]. A search for a solution should be conducted within a large space of possible mappings. The exact algorithms which provide optimal solution often take a long time to produce the optimal solution and therefore, in practice, an approximate algorithm is employed to deliver a near optimal solution in reasonable computation time.

3.1 Objectives

Although VM Placement has addressed in literature with subject to a variety of objectives from different perspectives, the most common objective is minimizing the number of active PMs. This is based on the underlying assumption that consumed energy is proportional to the number of powered-on PMs in a data center. Reducing the number of active PMs also contributes to the reduction of server footprint and capital investment in a data center [25]. However, VM placement can have other objectives such as power consumption or inter-communication among a set of VMs. A list of different objectives for VM Placement based on the literature is presented in Tables 1 and 2. In general, VM Placement is defined with a single objective or multiple objectives as shown in Fig. 4. As the name suggests, the single objective VM Placement is optimization (maximization or minimization) problem of one objective. In the multi-objective form of VM placement, two or more objectives are to be optimized simultaneously. For instance, minimizing the resource wastage in physical machines along with minimizing the power consumption is a multi-objective virtual machine placement problem with two objectives.

3.2 Constraints

Besides objectives, the search space for VM Placement can be restricted when constraints are introduced (as shown in Fig. 4). These constraints can be in two types: the basic constraints which actually serve as an intrinsic assumption to the problem and additional technical constraints which are added in accordance with a practical application or specific requirement.

In the following, the basic constraint and additional constraints are listed.

3.2.1 Basic constraints

- I: Each VM can be hosted by exactly one PM:

$$\sum_{j=1}^{n} x_{ij} = 1, \quad \forall i : 1 \leq i \leq m \tag{1}$$

  where $x_{ij} \in \{0, 1\}, 1 \leq i \leq m, 1 \leq j \leq n$ is 1 if VM i is assigned to PM j and 0 otherwise.
- II: For each type of resource (e.g. CPU, memory) and for each active PM, the sum of the resource demands for all the VMs sharing that PM should not exceed the capacity of the PM:

$$\sum_{i=1}^{m} d_{ri} x_{ij} \leq c_{rj} y_j, \quad \forall j : 1 \leq j \leq n \tag{2}$$

  where d_{ri} is the resource demand of type r by VM i, c_{rj} is the capacity of resource type r offered by PM j and y_j is 1 if PM j is active/powered-on and 0 otherwise.

3.2.2 Additional constraints

In addition to the above basic constraints, additional constraints may also apply. These constraints are as follows [45]:

- III: the number of VMs assigned to a particular PM j is limited to a certain number:

$$\sum_{i=1}^{m} x_{ij} \leq t_j, \quad \forall j : 1 \leq j \leq n \tag{3}$$

  where t_j is the maximum number of VMs, PM j can host.
- IV: A subset of all VMs; S may need to be assigned to the different PMs for security or technical purposes:

Table 1 Summary of different single objective VM placement schemes

Objective	Scheme	Assumption/drawback
• Number of PMs	[47]	• Only memory considered, the other resource types are overlooked
	[48]	• Resources other than CPU and memory neglected
	[51]	• For the sake of computational simplicity, other resources than CPU and memory are ignored
• Power consumption for PMs	[52]	• MBFD cannot always produce an optimal solution
	[62]	• SLA can be violated because of workload variability
	[64]	• BFD cannot guarantee the optimal solution
	[65]	• The average computation time of ELMS-ONC is significantly higher than that of FFD and OpenNebula scheduler
		• Complexity of ELMS-ONC
	[87]	• Lack of robust power model
	[63]	• FBFD cannot necessarily provide the optimal solution
	[68]	• The proposed method runtime performance is poor
	[71]	• The impact of other resources such as memory on energy consumption was neglected
• Power consumption for network communication	[73]	• Inter-VM traffic should be predicted accurately.
		• Scalability issue
	[74]	• The proposed strategy is limited to a single multi-core server
		• The proposed approach does not work with VMs more than number of cores
	[78]	• Only Fat-tree topology is considered in the simulation
• Satisfaction metric [79]	[79]	• The random placement used for comparison
		• A priori knowledge of communication patterns and flow demand profiles are assumed
		• VM inter-traffic is assumed to be negligible
• Traffic	[75]	• The time complexity of $O(n^4)$ for each recursive call
• Balance of residual resource usage/utilization	[82]	• The functionality of the suggested resource balance model is mainly influenced by the different parameters setting (i.e. $b_1, b_2\ \gamma_1, \gamma_2, \gamma_3$)
	[83]	• The issue of performance interference among different workloads (i.e. CPU-intensive, memory-intensive I/O-intensive) was not addressed.
• Resource usage	[80]	• Computational time analysis for the proposed scheme was not provided
		• The performance validation mainly focuses on the impact of different types of data center topology
• Carbon footprint	[86]	• Does not address the dynamic scenario
		• Lacks a rigorous and elaborative model to quantify the carbon footprint rate in a data center
		• Since the PUE metric does not consider the resource utilization the high value of PUE cannot guarantee the global efficiency in a data center
	[87]	• Lack of carbon footprint model
• Performance	[90]	• Only addresses the static scenario where the number of VMs is fixed
		• Only considers the static pricing schemas
		• No clear formulation of VM performance provided
		• In modern data centers moving toward fixed performance guarantee as SLA, the performance is not a crucial metric to be minimized [62]
	[40]	• The proposed performance model relies on cloud user to provide information in advance
		• In modern data centers moving toward fixed performance guarantee as SLA, the performance is not a crucial metric to be minimized [62]
• Cost of deployment	[40]	• When a priori knowledge of price is not available, a precise prediction tool is required

$$\sum_{i \in S} x_{ij} \leq 1, \quad \forall j : 1 \leq j \leq n \tag{4}$$

- V: A subset of VMs; S may need to be assigned to the same PM in order to facilitate inter-application communication or other requirements:

Table 2 Summary of multi-objective VM placement schemes

Objectives	Scheme	Assumption/drawbacks
• Number of PMs • Resource utilization	[69]	• The performance of the algorithm on minimizing resource utilization was not assessed • Lacks analysis of computational complexity of *ACO-VMP* over FFD
• Power consumption • Resource wastage	[95]	• No comparison with state-of-the-art Multi-objective algorithms (e.g. NSGA-II) presented • The computational complexity of *VMPACS* was not evaluated
	[42]	• No comparison with state-of-the-art multi-objective algorithms (e.g. NSGA-II) presented
	[103]	• The computation time of ICA was not analyzed in comparison to Genetic algorithm, ant colony and FFD placement approaches
• Power consumption • Performance (response time) of VMs	[65]	• No comparison with state-of-the-art multi-objective algorithm (e.g. NSGA-II) presented
	[9]	• Homogenous PMs • Only CPU-intensive and Memory-intensive workloads considered (does not include traffic intensive and disk-intensive workloads) • There is no comparison for computational time. • In modern data centers moving toward fixed performance guarantee as SLA, the performance is not a crucial metric to be minimized [62]
• Power consumption for PMs • Power consumption for network elements	[107]	• Only applicable to the multi-tier tree topology
• Traffic amount VMs • Number of PMs	[39]	• Prior knowledge about inter-connection structure is required • Assumes only layered structure for application
• Resource wastage • Power consumption • Thermal dissipation	[81]	• The large computation time of GA
• Power consumption • Interference among VMs	[105]	• High runtime cost of SA technique
• Resource usage • Server usage • Bandwidth usage	[44]	• Weighted sum approach's drawback • High time complexity $O(N^2M^2)$ N = Number of PMs, M = Number of VMs
• Power consumption • Network traffic • Migration cost	[106]	• Weighted sum approach drawback • A priori knowledge of the traffic matrix is required • Aggregated network traffic in the same link incurs congestion • Poor stability of *VM-Mig* for dynamic placement • *VM-Mig* is prone to get stuck in local optimum
• Number of PMS • Resource wastage	[7]	• Determining parameters Satisfaction factor and balance factor is difficult for the normal user without expertise
• Maximum bandwidth occupancy on the uplink of all the ToR switches • Maximum number of VM partitions of all the requests	[121]	• Only tree-like topology is considered. • The locality studied in ToR switch level only
• Power consumption • Performance degradation	[59]	• The impact of other parameters than CPU in estimating performance degradation neglected
	[113]	• Comparison of PPVMP is made with other methods that do not consider performance degradation as objective
• Response time • Failure rate • Resource utilization	[108]	• The ability of the proposed algorithm to provide the optimal or sub-optimal solution to the problem was not validated. • A relative high computational complexity ($O(a \times v \times n)$)

Single objective form	
Minimize	
Subject to	*Constraint* ***1****, Constraint* ***2****, …,* Constraint ***k***
Multi-objective form	
Minimize	*Objective* ***1****, Objective* ***2****, …, Objective* ***n***
Subject to	*Constraint* ***1****, Constraint* ***2****, …,* Constraint ***k***

Fig. 4 A general formulation of the VM placement problem

$$\sum_{i \in S-\{e\}} x_{ij} = (|S| - 1) \cdot x_{ej}, \quad e \in S, \forall j : 1 \leq j \leq n \tag{5}$$

- VI: A particular VM v may need to be assigned to a subset; R of PMs. This is because v requires a certain hardware specification such as storage or network bandwidth that only provided by PMs in R:

$$\sum_{j \in R} x_{ij} = 1, \quad \forall i : 1 \leq i \leq m \tag{6}$$

- Also, any problems objective can also act as a constraint if it does not appear as an objective. For example, minimizing the number of PMs while satisfying the inter-traffic among VMs.

3.3 Solution

A potential solution to the VM placement problem with m VMs and n PMs can be represented by the matrix shown in Fig. 5. However, among potential solutions, only solutions that satisfy the basic constraint are feasible solutions. As it can be inferred from the matrix in Fig. 5, the number of the different combination is $2^{m \times n}$ as each cell takes either 0 or 1. In fact, $2^{m \times n}$ is the size of search apace for a typical VM placement problem and it is also binomial time complexity of a brute-force algorithm that enumerates all the possible solutions in the search space to find the optimal one. This can be an indication of the intractable nature of the VM placement problem.

List of VMs \ List of PMs	PM_1	PM_2	…	PM_n
VM_1	x_{11}	x_{12}	…	x_{1n}
VM_2	x_{21}	x_{22}	…	x_{2n}
⋮	…	…	…	…
VM_m	x_{m1}	x_{m1}	…	x_{mn}

Fig. 5 Matrix representation of potential solutions to the VM placement problem

4 State-of-the-art VM placement strategies

In this section, the state-of-the-art VM placement approaches are classified into two categories, namely: single-objective and multi-objective approaches. Specifically, for each category, a number of salient methods in the literature are described according to the particular objective used.

4.1 Single objective VM placement

In the following, a number of eminent works that attempted to address a single objective VM are reviewed according to their specific objectives. Finally, at the end of this section, a summary of discussed schemes together with their corresponding objectives and their assumption/drawback is presented in Table 1.

4.1.1 Number of PMs

One of the common and intuitive approaches for reducing power consumption is through minimizing the number of PMs in a data center. In such approaches, power consumption is deemed to be proportional to the number of PMs [7]. This objective can be represented by the following formula:

$$minimize \sum_{j=1}^{n} y_j \quad \forall i : 1 \leq i \leq m \tag{7}$$

where y_j is 1 if PM j is active/powered-on and 0 otherwise. Figure 6 shows an example of minimizing the number of PMs in a data center. The upper part is a data center with five underutilized PMs whereas the bottom part is the same data center with a minimum of two highly utilized active PMs and three turned off servers. The load on three switched off PMs has been transferred to the active PMs.

Tang et al. [47] proposed a dynamic forecast scheduling algorithm called VM-DFS for VM placement. The problem is formalized as bin packing problems and FFD algorithm is employed to solve the problem with the objective of minimizing the number of active PMs. VM-DFS uses a prediction model to forecast the future memory consumption of VMs with dynamic memory demand and place VMs on the most suitable PMs based on their predicted future consumption. The result of the simulative experiment in *CloudSim* shows that VM-DFS reduces the number of active PMs as compared to the default static algorithm in *CloudSim.* However, the proposed algorithm merely considers the memory usage and the impact of other important resources like CPU and bandwidth are overlooked.

Liu et al. [48] developed an Ant Colony System (ACS) based algorithm named OEMACS coupled with a local search technique to minimize the number of active PMs in

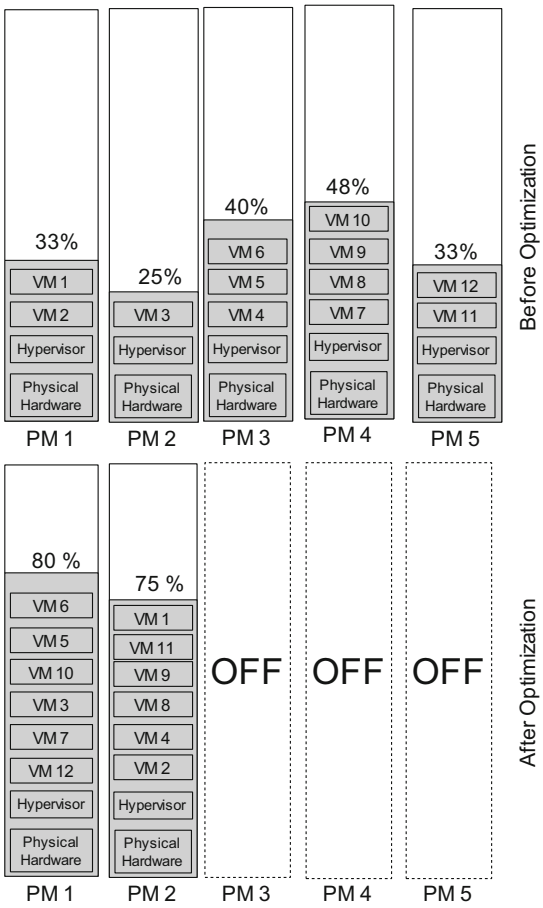

Fig. 6 Minimizing the number of PMs

a cloud data center. The problem formulation merely considers two types of resources as CPU and memory. The performance of OEMACS assessed by comparing the results to that of FFD [49], RGGA [50], ACO and MACO obtained from a series of experimental tests with both homogeneous and heterogeneous servers. The results indicate that OEMACS can find an optimal or quasi-optimal solution in both homogeneous and heterogeneous data center environments in terms of a number of active servers and average memory and CPU utilization.

Yan et al. [51] presented SOWO, a discrete particle swarm optimization (PSO) based VM placement algorithm to minimize the number of active PMs in the cloud center. To reduce the computational complexity of the problem, SOWO only focuses on CPU and memory as the two most critical system resources. For each PM, total workload for placing k VMs on a that PM defined as below formula:

$$\frac{1}{1-\left(cpu_p+\sum_{i=1}^{k} cpu_i\right)} \times \frac{1}{1-\left(mem_p+\sum_{i=1}^{k} mem_i\right)} \tag{8}$$

where cpu_p and mem_p are the current CPU and memory load of the PM respectively. Also, cpu_i and mem_i are CPU utilization and memory utilization of VM V_i. The authors used the presented workload formulation as a fitness function for their PSO-based algorithm.

SOWO was implemented as a scheduler in OpenStack and the experiment was conducted to compare the usability and superiority of SOWO compared to the native OpenStack scheduler known as *Filter Scheduler*. In the experiment, ten homogenous PMs with identical specifications were used along with four types of VM templates. The result of experiments shows that SOWO is capable of using fewer PMs compared to native OpenStack native scheduler. Also, in terms of computational time, the result indicates two methods behave almost similar when the number of input VMs increases. However, when it comes to resource utilization, OpenStack native scheduler performs more stable and balanced than SOWO.

4.1.2 Power consumption

Power consumption of a server in a data center is often determined by the sum of power consumption of all the main hardware components in that server. The components are CPU, memory, disk storage, and network adapter. However, CPU consumes the largest fraction of the energy as compared to other hardware parts [52]. Recent studies [42, 52–55] report that there is a linear relationship between the power consumption of a server and its CPU utilization level. Furthermore, a server in its idle state still consumes 70% of the power when it operates with maximum capacity. Hence, power consumption is usually defined as a function of CPU utilization as shown in the following formula [30, 41, 52, 56–59]:

$$P(U) = P_{idle} + \left(P_{busy} - P_{idle}\right) \times U \tag{9}$$

where P_{idle} is the power consumption in idle state, P_{busy} denotes the maximum power consumption when the CPU is fully utilized, U is the CPU utilization of server and $P(U)$ is the power consumption of the server based on its utilization level. The CPU utilization is variable over time due to change in CPU load and therefore the total energy consumption (E) for a period of time; $[t_a, t_b]$ is calculated as follows [52, 60]:

$$E = \int_{t_a}^{t_b} P(U(t))dt \tag{10}$$

It is noteworthy that Dynamic Voltage and Frequency Scaling (DVFS) is an effective technology in managing the energy consumption of the processor. This technology allows the processor to operate in variable frequencies with different voltages. However, although DVFS has been

widely applied in embedded, multicore and multiprocessor systems, it is less adopted in virtualized data centers [58]. As a result, the servers in today data centers are not energy proportional [61].

Verma et al. [62] presented design, implementation, and evaluation of placement controller called *PMapper* to address application placement and VM placement for a heterogeneous data center. The VM placement is performed in the second phase and it is intended to minimize the power consumption under a fixed performance constraint in the form of SLA. *PMapper* uses an extension of FFD heuristic (called Min Power Parity or mPP) in which more power-efficient servers are utilized first.

Beloglazov et al. [52] presented a modified version of Best-Fit Decreasing heuristic called MBFD to solve the static form of VM placement. In MBFD, all VMs are sorted in descending order of their CPU demands and each VM is mapped to the PM with sufficient capacity and with least increase of power consumption after this mapping. The time complexity of the algorithm is $O(n.m)$ where n is the number of VMs and m is the number of PMs. Abdullah et al. [63] enhanced MBFD by proposing fast best-fit decreasing (FBFD). The main idea of FBFD is to sort the list of PMs in increasing order of CPU utilization of PMs using a binary search tree before the placement takes place. After each placement, the list of PMs is sorted again. As a result, the VMs are assigned to a PM with most power-efficient PMs first. As FBFD uses a binary search tree to find the most suitable PM with time complexity of $O\left(log_2^n\right)$ instead of searching the whole list in MBFD, its overall time complexity is $O\left(m \cdot log_2^n\right)$ as compared to $O(m \cdot n)$ of MBFD. In the dynamic scenario and to determine when to migrate VMs, a double-threshold policy introduced. The CPU utilization level of PM should be always between a lower threshold (T_l) and an upper threshold (T_u). If the total CPU utilization level of a PM exceeds T_u, some VMs have to migrate to other PMs. In the event that the CPU utilization of PM falls below the T_l, all the VMs on that particular PM have to migrate to other PM in order to reduce the utilization level and prevent performance degradation. The authors also proposed three different policies to determine what VMs should be migrated. These policies are *Minimization of Migration (MM)* policy, The *Highest Potential Growth (HPG)* policy, and *Random Choice (RC)* policy. MM policy identifies the minimum number of VMs to be migrated according to the current utilization level of PM along with the two thresholds. RC policy randomly selects a number of VMs to be migrated. HPG migrates a set of VMs with the lowest usage of CPU relatively to the CPU capacity as defined by set S in the following formula:

$$\mathrm{S}|S \in P\left(V_j\right), u_j - \sum_{v \in S} u_a(v) < T_u, \sum_{v \in S} \frac{u_a(v)}{u_r(v)} \rightarrow \min \qquad (11)$$

where V_j is the set of VMs already placed on PM j. $P(V_j)$ is the power set of V_j, u_j is the current CPU utilization of PM j, $u_a(v)$ is the CPU utilization allocated to VM v and $u_r(v)$ is the CPU initially requested by VM v. The simulation result shows that, overall, using MM policy provides the best results among other policies in terms of SLA violation, energy saving and the number of VM migration.

Gao et al. [64] presented a dynamic resource management scheme to minimize the power consumed by the physical infrastructure while SLA requirement is also met. The proposed scheme leverages both DVFS and server consolidation to reduce power consumption and provides desired performance guarantee. In particular, a greedy heuristic on the basis of BFD was presented to assign a set of VMs to PMs with regards to the above-mentioned objective. The result of experimental validation shows that the proposed scheme yields 50.3% power saving as compared to the four other policies.

Kessaci et al. [65] proposed a new placement technique to be embedded in OpenNebula [66]; a cloud management software. The proposed algorithm called *EMLS-NOC* is based on multi-start local search metaheuristic and aims to minimize the energy consumption of the entire infrastructure. The multi-start feature is used to give more exploration power within the search space to the algorithm while local search adds more accuracy to the algorithm. *EMLS-ONC* was compared to OpenNebula's default scheduler and *FFD*. The result shows *EMLS-ONC* improves the OpenNebula's default scheduler 26% on average and it also improves energy-aware FFD-based approach up to 25%.

The dynamic form of VM placement also was addressed by Ferreto et al. [67]. In particular, the authors investigated mapping VMs with variable resource demands into the smallest number of PMs. As opposed to the static form where the VM resource demands remain unchanged during VM lifetime and VM resource capacities are assigned according to a peak demand of VM, VMs with dynamic demands, changes their resource demands based on their current and actual need. However, the dynamic approach might result in migrating VM between different PMs in order to remove VMs from an overloaded PM or to switch off a PM when all its VMs already moved to other PM. The authors presented a Linear Programming (LP) formulation to deal with the problem. The whole lifetime of VM is divided into a number of consolidation steps. In each consolidation step, the placement process is repeated using VM demands at that particular moment which may require migration of VMs to different PMs and also can affect the number of required PMs. The key idea behind the proposed

approach (called *dynamic consolidation with migration control*) is to avoid migrating VMs with steady demands. Performance degradation due to migration or QoS deterioration was stated as justification for prioritizing the VMs with fixed demands. The authors also used the same idea to some common heuristics as FFD, BFD, WFD, and AWFD and made following modifications in order to ensure VMs with steady demands are not migrated: (a) in each consolidation step, map VMs with steady demands to the previously mapped PMs. (b) sort physical servers according to the lexicographic order of their resource (CPU, memory, and network) capacity. The proposed approach was evaluated based on TU-Berlin and Google workloads and the main finding is: avoiding migration of VMs with steady demands reduces the total migration while it has a minimal adverse effect on the number of PMs.

In another study, Alharbi et al. [68] took advantage of ant colony system optimization technique to solve the dynamic VM placement problem with the objective of minimizing the total energy consumption of all active PMs in a data center. The energy consumption is modeled similar to what we presented earlier in Eq. (9). The proposed method utilizes the VM and PM profile information extracted from the historical data logs. This information includes CPU and memory capacity, their residual capacity and their minimum and maximum energy consumption. The dynamic scenario is implemented considering a change in VM request in each time interval. At the beginning of each interval, the list of VMs is updated and the released resource by expired VMs are made available to the new VMs. According to the conducted simulation on small, medium, and large-scale data centers, the proposed method offers improved energy efficiency in comparison with FFD, ACO-VMP [69] and PVM [70]. However, such an improvement gained at the cost of more execution time. In terms of scalability, the runtime of the proposed method shows a linear increase with varying the number of PMs.

To minimize the overall energy consumption in a data center, Xiao, Ming [71] proposed a partitioned optimization framework. The proposed framework classifies the set of PMs into three different pools, namely: *running pool*, *sleeping pool* and *off the pool*. The running PMs are currently loaded with VMs while off PMs are switched off for energy efficiency. The sleeping PMs are put into the low energy state and they can be awakened when they are needed. The PM in different pool consumes a different amount of energy. The overall energy model which is denoted by E_{all} is defined as a summation of three parts as it is shown by the below equations:

$$E_{all} = \sum_{i=1}^{m} (E_{sta}(i,t) + E_{swi}(i)) + \sum_{j=1}^{n} E_{mig}(j) \quad (12)$$

where $E_{sta}(i,t)$ is the energy consumption of ith PM in its state (running, sleep or off). $E_{swi}(i)$ is the amount of energy consumed for state switching of ith PM. $E_{mig}(j)$ is the energy consumption for migrating jth VM. The main idea of the proposed optimization method is to reduce the search space by avoiding states/solutions that cannot make the current energy consumption any lower. The authors proposed a memetic algorithm to solve the dynamic placement of VMs. To demonstrate the priority of the proposed algorithm, the experiment was conducted to compare the proposed algorithm to the heuristics: FF(First Fit), BFI (Best Fit Increasing), BFD (Best Fit Decreasing), Greedy and LB (Load Balance). Based on the results, the authors conclude that the proposed algorithm outperforms those heuristics algorithms in terms of the percentage of energy consumption improvement. One drawback of the proposed method can be ignoring the impact of other hardware resources such as memory and even GPU on the energy consumption in data centers. In fact, the inclusion of such resources in the energy model can help more precise prediction of energy consumption and therefore the more realistic solution for the VM placement.

4.1.3 Power consumption for network communication

Most studies on efficient power management in the data center primarily focus on the effects of computer hardware with high power demands such as computer servers and cooling systems. However, network equipment also consumes 10–20% of total power in a data center [72] which introduces a new challenge to reduce the networking power consumption without imposing an adverse impact on overall network performance. This issue motivated Fang et al. [73] to propose a novel approach called *VMPlanner* to conserve network power in a data center. The main idea behind *VMPlanner* is to optimize the VM placement and traffic flow among VMs such that dispensable networking elements can be switched off for the sake of maximum possible energy conservation. The problem is formulated as a combinatorial optimization problem and solved in three different steps. First, all the VMs are partitioned into a set of groups with a minimum amount of inter-communication. Second, using a Tabu search technique, VM groups are assigned to the corresponding PMs' racks such that total inter-rack communication is minimized. Third, the network traffic among VMs is managed such that dispensable networking equipment can be switched off for energy conservation. The evaluation result shows *VMPlanner* achieves 60% more power saving as compared to the situation in which all the network equipment are fully operating.

In multi-core servers, the inter-communication among the cores can substantially impact the overall system performance. Network-on-chip (NOC) technology leverages the computer networking and packet switching concept to provide efficient communication among multiple cores compared to the conventional communication architecture which uses wires and buses to connect cores. The advantage of NOC over traditional architecture is low latency, high performance, and lower power consumption. Liu et al. [74] designed an energy-aware on-chip VM placement scheme with the aim of efficiently allocating a number of VMs on a multi-core server with high efficiency and performance. An ant colony heuristic is used to place the VMs running the same application on the closer cores based on traffic rates, energy consumption, and communication delay. The problem is formulated as binary integer programming with the objective of minimizing the power consumption for inter-communication between VMs. The simulation results show that the proposed scheme attains better energy efficiency compared to FFD placement and random placement.

4.1.4 Traffic

A typical topology for a data center is a tree structure which includes switches. In such a topology, the communication delay between two different VMs is proportional to the distance between them. The distance is a number of hops from sender VM to the receiver VM. When a collection of VMs forms a single application, an inter-communication among collaborating VMs is likely needed. Therefore, placing the most communicative VMs on PMs with minimum network distance is a way to alleviate the communication overhead [39]. Besides VM placement policies to minimize the number of active PMs, power consumption or other criteria, *network-aware* techniques [39, 75] intend to improve the performance of the application by minimizing the communication latency among VMs. However, in such techniques, clients are required to at least provide the application interconnection network and its communication requirement in order to facilitate the decision-making process in effective resource management. As a simple example shown in Fig. 7, the placement strategy can accelerate data transfer by moving the communicative VMs (e.g. VM1–VM3, VM2–VM4) from distant physical servers to a local physical server.

Meng et al. [75] addressed the scalability concern for modern traffic-intensive data centers. The proposed traffic-aware placement policy (called *Cluster-and-Cut*) intends to minimize the average traffic latency for the data center network by placing most communicative VMs in close proximity. *Cluster-and-Cut* is a two-tier heuristic algorithm which receives the traffic matrix between the VMs and first

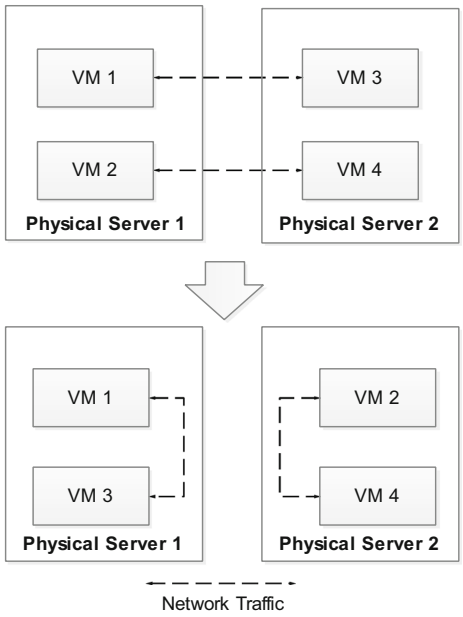

Fig. 7 Network traffic optimization

partitions VMs and hosts into different clusters and then matches VMs to the hosts at cluster level and thereafter at the individual level. The experimental analysis indicates that the proposed placement algorithm significantly reduces the aggregate traffic and computational time as compared to the existing generic methods presented in [76] and [77]. In the other effort to minimize the energy consumption of network equipment, da Silva, da Fonseca [78] presented a topology-aware strategy to place communicating groups of VMs closer together in a small area of the data center. As a result, VMs require a shorter path and fewer network switches to communicate with each other thus less energy consumed. The proposed algorithm, called *TAVMP*, receives a group of VMs as input and then splits the whole data center topology into smaller sub-graphs. The same strategy is recursively applied to each sub-graph and when the lowest level is reached the placement decision is made by other algorithm named Placement in Current Area (*PCA*). The performance of *TAVMP* was assessed in a simulation experiment based on blocking ratio (percentage of VMs were not placed) and energy efficiency. The result indicates that *TAVMP* accepts more virtual machines without degrading the energy efficiency compared to other algorithms: Power Aware Best Fit Decreasing algorithm (*PABFD*) and Round Robin algorithm (*ROUND*).

Rahimzadeh Ilkhechi et al. [79] studied VM placement with the objective of maximizing a certain metric named *Satisfaction* in a particular scenario of interest where some VMs are highly inclined to exchange traffic to certain nodes called sinks. The sinks can be a supercomputer,

connection point or any physical resource that other nodes are highly dependent on it. The *satisfaction* of a VM is measured based on appropriateness of a PM that hosts that VM. Moreover, the metric takes into account the cost (proximity) of VMs to sinks together with demand flow of VMs in order to determine the suitability of each PM. The authors presented greedy and heuristic-based algorithms to assign VMs to PMs and found these algorithms more effective compared to the random assignment. However, the presented algorithm assumes that the knowledge of communication patterns and flow demand profiles are provided beforehand.

Song et al. [80] formulated the VM placement problem as a convex optimization problem and proposed a scheme called optimization-based scheme for solving the problem in a large scale data centers which take into account both network dependencies between different VMs and server constraints. The problem objective is to minimize the communication traffic among VMs. To validate the performance of the proposed approach, it was compared to the random placement and traditional bin packing algorithm in four different scenarios and with subject to four popular data center architecture topology such as *Tree*, *VL2*, *Fat-Tree,* and *BCube*. The achieved results indicate that employing the optimization-based schemes in such topologies (Especially *BCube*) reduces the communication cost between VMs and thus results in a higher degree of performance as compared to other methods. In addition, the proposed scheme requires the least number of PMs as compared to the random placement and First Fit placement.

4.1.5 Balance of the residual resource/resource utilization

The residual resource along different dimensions on each server should be always balanced in anticipation of future request. This is to prevent any resource wastage due to fragmentation [81]. Figure 8 shows an example of resource allocation along two dimensions; namely CPU and memory for a typical PM.

Figure 8a shows an unbalanced placement strategy which results in resource wastage while in Fig. 8b the balanced placement helps to provide sufficient capacity to the future requests. The resource wastage in Fig. 8a is caused because the remaining resource along CPU dimension is too small and thus unlikely to accommodate future requests. By placing each VM on a PM, a certain amount of resource in different dimensions on that particular PM is consumed. Each inner rectangle represents the resource usage of each VM while the outer rectangle is the total resource capacity of the PM. Some extant works [7, 82] addressed VM placement with load balancing as an objective. Specifically, Cho et al. [82] proposed a hybrid meta-heuristic called *ACOPS* (as a hybrid of ant colony optimization and particle swarm optimization) to maximize the balance of resource utilization across different resource dimensions. The proposed approach uses the workload of historical requests to predict the workload of future requests. Each request is for a VM along with its resource demands. To quantify the load balancing, a degree of balance (*DB*) is defined for three types of resources i.e. memory, CPU and disk as follows:

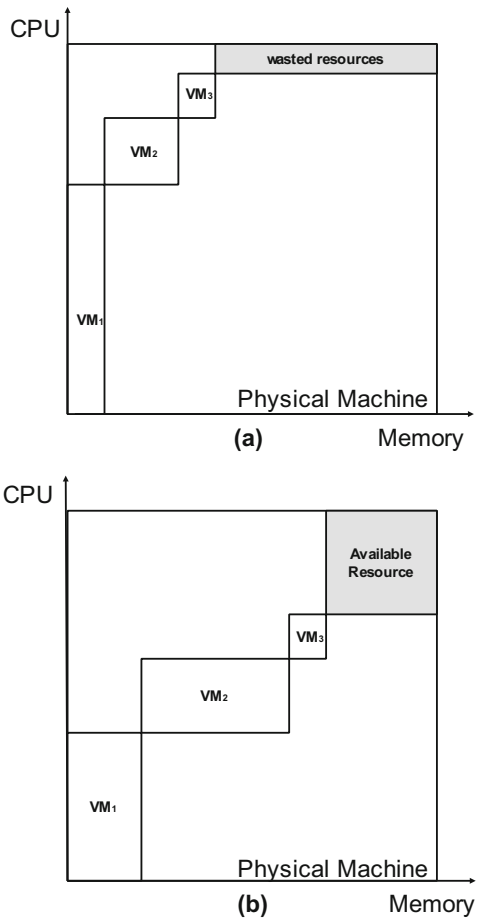

Fig. 8 **a** Unbalanced PM versus, **b** balanced PM

$$DB = b_1 \times V + b_2 \times (1 - D) \tag{13}$$

where V is the feature of utilization, D is the feature of maximum difference. b_1, b_2 are coefficients. In addition, to speed up the *ACO* process, the unsatisfied solutions are rejected before scheduling. The results of the simulative experiment indicate that *ACOPS* is faster than conventional ant colony optimization algorithm, has shorter makespan and also outperforms other approaches in terms of the balance of resource utilization. The authors also reported the time complexity of the proposed algorithm as $O(n^{2}\text{-}MAI)$ (n number of VMs, M number of PMs, A number of ants, I maximum number of algorithm iterations).

He et al. [83] developed a Genetic Algorithm (GA) to consolidate moldable VMs for a cloud system. In converse to rigid VMs that the resource capacities remain unchanged, the resource capacity for moldable VMs is adjustable. Their approach particularly deals with a set of virtual clusters each of which provides a specific type of service to the users. Since a steady level of Quality of Service (QoS) is expected from the whole cluster as a single entity, it is not necessary to keep individual VMs capacities constant in the cluster. The authors employed a genetic algorithm to search for a mapping solution that most minimizes the standard variation of spare capacity across different resource types. The outcome of the genetic algorithm is an optimized system state that represents the optimal mapping of VMs in a virtual cluster to the PMs. If the resource capacities of VMs undergo a change, a new system state is calculated and the old state is transited to the new one. Transiting to the new state may involve different VM operations as VM creation, VM deletion, and VM migration. As each of VM operations has different cost, a heuristic approach was developed to obtain a reconfiguration plan with lowest possible cost in a reasonable time. In addition to the initial placement of VMs, a reconfiguration algorithm is leveraged to dynamically transform the current state of allocation to the new one. Through a simulation experiment, the developed GA technique was compared to the Entropy consolidation scheme presented in [84] and the result demonstrates that *GA* performs better than Entropy in packing VM in a fewer number of physical nodes.

4.1.6 Carbon footprint

The amount of carbon dioxide gas (CO_2) emitted from an energy source of a data center is referred to as ICT carbon footprint and considered as an acute environmental effect. Today's, large scale data centers are confronted with a substantial increase in carbon emission and therefore minimizing the carbon footprint has become one of the significant industry priorities [85]. Proper handling of the issue will contribute towards a sustainable and green ICT technology. Khosravi et al. [86] proposed an Energy and Carbon-Efficient VM placement algorithm called *ECE* based on a best-fit heuristic. *ECE* places VMs on a distributed data center with the objective of minimizing the carbon footprint. A broker decides to place the VMs on most suitable sites and servers according to the different parameters such as data center power usage effectiveness (*PUE*), energy source carbon footprint rate and proportional power. *PUE* is a metric to measure data center efficiency. A data center *PUE* is calculated as:

$$PUE = \frac{Total\ date\ center\ power\ consumption}{Data\ center\ IT\ power\ consumption} \quad (14)$$

where *total data center power consumption* refers to the sum of power drawn by the data center for all the purposes including IT equipment, lightning, cooling ant, etc. *Data center IT power consumption* reflects the power consumed by the data center for IT equipment only (as illustrated in Fig. 9).

PUE is a value greater than 1. In an ideal condition, PUE = 1 implies that 100% of electricity provided to a data center goes to the IT equipment which is practically impossible. The smaller the *PUE* is the more efficient data center is. The higher values of *PUE* mean a larger portion of input electricity is spent on cooling, lighting and etc. To evaluate *ECE*, it was compared with four First-Fit based heuristics in four data centers with heterogeneous infrastructure. The results demonstrate that, with the increasing number of VMs, *ECE* reduces the carbon footprint by at least 45%. Moreover, In terms of power consumption, *ECE* achieves a minimum of 8% of power saving.

Moghaddam et al. [87] proposed two new algorithms for placement of VMs in multiple clouds. The algorithms are an extension of the *GGA* algorithm presented in [81] and these algorithms were studied with different objectives such as minimizing energy consumption (*MLGGA-EA*) and carbon emission reduction (*MLGGA-CA*). The proposed algorithms are compared to *GGA*, *FFD* and Swarm [88]. The result demonstrates that, overall, *MLGGA-EA* achieves better solutions for multi-cloud scenario while *MLGGA-CA* is a promising choice for energy efficiency case. However, the *MLGGA-EA* is not recommended when carbon footprint is the main concern. The proposed approach is devoid of any concrete formulated model for energy consumption and carbon footprint.

4.1.7 Total VM performance

Service Level Agreement (SLA) is an agreement between user and provider and specifies a minimum level of quality

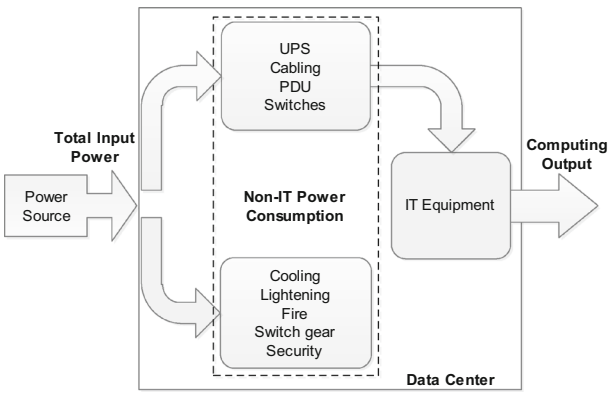

Fig. 9 Illustration of different power consumption for calculating PUE

of service to be offered to the user [89]. In the IaaS cloud model, one of the important factors to comply with the endorsed SLA is to ensure tenants always receive the promised quality and specification for the hardware equipment they lease. In particular, in the presence of virtualization technology and with interference caused due to co-existence of several VMs on a single PM, an important requirement to be taken into account is to improve VM performance which can be represented as VM response time (delay) [65] or VM throughputs.Tordsson et al. [90] presented a cloud brokering approach that involves optimal placement of VMs across multiple heterogeneous clouds. The cloud broker has two roles, First: providing a scheduling mechanism for determining optimal VM placement. Second: providing a uniform and transparent management interface for dealing with different VMs without depending on a specific type of cloud architecture or technology. A schematic of architecture for proposed cloud brokering approaches is shown in Fig. 10. The placement algorithm is in a static form and designed based on binary Integer Programming formulation and meant to maximize the total performance of running VMs across multiple clouds while satisfying various constraints such as performance, budget, service configuration and load balancing. The objective function is represented as:

$$TIC = \sum_{j=1}^{l} C_j \left(\sum_{i=1}^{n} \sum_{k=1}^{m} x_{ijk} \right) \tag{15}$$

where C_j is the performance of a VM type j and $x_{ijk} = 1$ if VM i of type j is placed on cloud k, and 0 otherwise.

To solve the problem, a mathematical programming language called *AMPL* [91] used along with a *CPLEX* [92] as a backend solver. The cloud brokering approach is evaluated with high throughput computing cluster over multiple cloud providers and the most significant finding is: the deployment of VMs over multi-cloud results in better performance and lower cost as compared to single cloud scenario.

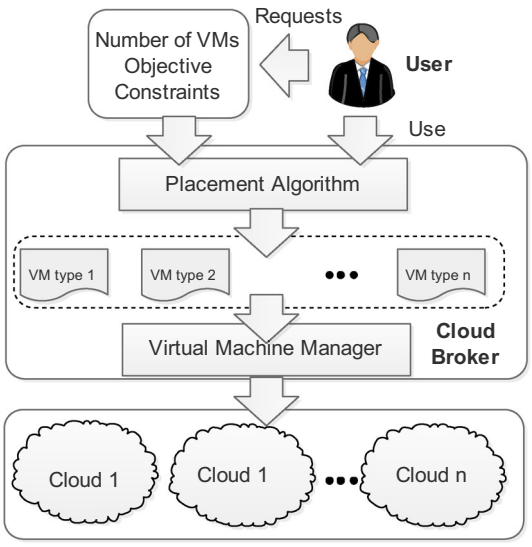

Fig. 10 Cloud brokering approach

4.1.8 Cost of deployment

In today cloud market, service providers offer a diverse range of service plans. These plans are sometimes subject to revision by the provider from time to time. However, it is not always straightforward for cloud end users to choose the best economically possible service with subject to their budget and limitations. This is particularly true in the case of the federated cloud that a service has to be provisioned through a set of stand-alone clouds with different pricing schemas. In this scenario, an intuitive goal is to minimize the total cost of deployment of VMs by choosing the lowest price plan/cloud. The cost is defined as a sum of the cost for each VM to be deployed [40]. Analogous to Tordsson et al. [90], Lucas-Simarro et al. [40] proposed a modular cloud brokering architecture including a scheduling module for the multi-cloud scenario. The VM placement strategy was implemented in the scheduling component and was aimed to optimally deploy VMs across different cloud environments where each vendor offers different and dynamic price schemes. The problem is formulated to minimize the so-called *Total Infrastructure Cost* (*TIC*). *TIC* is defined as a total cost for placing each VMs for a particular period of time as represented as:

$$TIC(t) = \sum_{i}^{n} \sum_{j}^{l} \sum_{k}^{m} X_{i,j,k}(t) * P_{j,k}(t) \tag{16}$$

where t is the 1-h period of time and $X_{i,j,k}(t) = 1$ if VM i of type j is placed on cloud k during period t, and 0 otherwise. $P_{j,k}(t)$ is the price of placing VM j on cloud k for a period t. The placement is repeated before the beginning of each 1-h period and the prices of similar VM instances are subject to change over periods of time. Additionally, the second objective is to maximize *Total Infrastructure performance (TIP)* as a total performance for each VM in a certain period of time.

$$TIP(t) = \sum_{i}^{n} \sum_{j}^{l} \sum_{k}^{m} X_{i,j,k}(t) * Perf_{j,k}(t) \tag{17}$$

where $Perf_{j,k}(t)$ is the performance of VM of type j on cloud k for a period t. The performance of a virtual machine depends on a number of factors as a requirement of the application, type of VM and the PM that hosts the VM. LINPACK benchmark [93] was used to analyze the performance of each instance of VM.

In addition, few constraints with respect to the budget, minimum expected performance and budget were added. Thereafter, *AMPL* [94], a mathematical programming language, with *MINOS* solvers is used to optimize the mathematical model. The performance of the architecture and placement strategy was evaluated against *HPC* cluster and Web Server cases and the results demonstrate that multiple VM placement outperforms single VM and multi-cloud deployment using the broker is superior to the single one regardless of interference of cloud broker. Also, making use of cloud broker, users are benefited with 4–6% improvement in performance or budget.

4.2 Multi-objective VM placement

This section is meant to review a number of prominent VM placement approaches that address the multi-objective form of the problem. Besides, at last, a summary of discussed schemes as well as their objectives and assumption/drawback is presented in Table 2. Although the majority of proposed VM placement techniques formulate the problem based on a single objective there exist some other strategies [12, 41, 65, 95] that mainly address the multi-objective variation of the problem. Despite the primary objective which is often minimizing total power consumption, other criterions like *Traffic*, *Load balancing*, and *Thermal dissipation* can be considered to establish the multi-objective form of the problem. In multi-objective optimization, the aim is to minimize/maximize the number of objectives simultaneously. Generally, in tackling multi-objective problems there are two main approaches: The first approach relies on Pareto concept [96] to find a range of tradeoff solutions which are equally optimal and called non-dominated solutions as shown in grey in Fig. 11. The figure illustrates a sample solution space for a problem of minimizing two objectives of power consumption and network traffic. The circles in white represent solutions that have greater (worse) values of both objectives and therefore dominated by better solution represented in grey. The second approach of multi-objective optimization transforms an originally multi-objective problem into a weighted sum of individual objectives as a single objective as shown in the following equation for two objectives of power consumption and traffic:

$$Minimize \;\; w_1 \times power + w_2 \times traffic \tag{18}$$

where w_1 and w_2 are weighting coefficients for power consumption and traffic respectively and they indicate the relative importance of their corresponding objectives. Although the weighted sum approach is regarded as the simplest way to deal with a multi-objective problem, choosing the proper weight vector is not always a straightforward task [97]. Moreover, prior determination of the weight vector excludes some other potentially good solutions from the search space [98]. In practice, the weighting coefficients are chosen based on the relative importance of individual objectives for the specific problem of interest [96]. For example, if in a situation, minimizing power usage has more priority than minimizing traffic overhead then in above $w_1 = 0.7, w_2 = 0.3$ can be a reasonable choice or in case of the equal importance of two objectives, both coefficients can be set to 1 (i.e. $w_1 = 1, w_2 = 1$). Table 3 lists VM placement schemes that utilize a weighted sum approach to deal with multiple objectives along with their rationale behind choosing weighting coefficients.

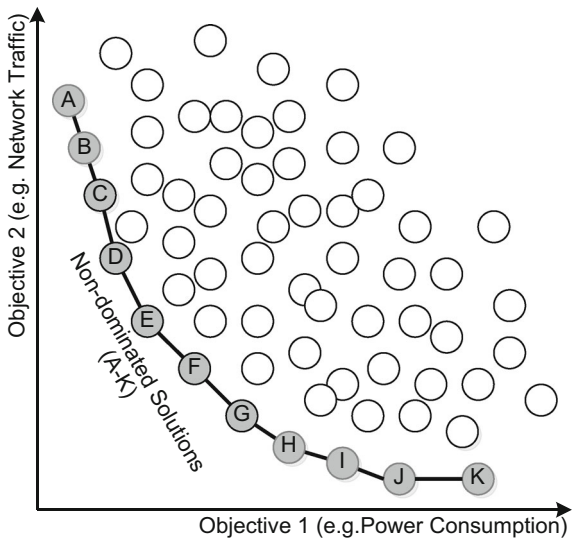

Fig. 11 Pareto concept for multi-objective optimization

4.2.1 Number of PMs and resource utilization

Liu et al. [69] proposed an ant colony based algorithm called *ACO-VMP*. The objective is to minimize the sum of total resource utilization and the number of used servers as shown by below formula:

$$w_1 \cdot \sum_{i-1}^{M} \left(\frac{1}{PC_i - UC_i} + \frac{1}{PM_i - UM_i} \right) + w_2 \cdot M \tag{19}$$

where w_1 and w_2 are weight coefficients, PC_i and PM_i represent the CPU and memory capacities respectively while UC_i and UM_i are CPU and memory utilization level respectively. M is the number of PMs. The performance of *ACO-VMP* was compared to *FFD* algorithm in [49] and the experimental results show the solution returned by *ACO-VMP* always requires fewer PMs when VMs are varied between 100 to 600. However, no comparison between the two algorithms was performed in terms of computational time. Also, the performance of *ACO-VMP* in minimizing resource utilization was not assessed.

Table 3 List of VM placement schemes with weighted sum approach

Scheme	Choice of weighting coefficient for multiple objectives
[69]	The authors used w_1 and w_2 to scalarize the *number of PMs* and *resource utilization* as shown by the below formula. However, they did not indicate how they determined specific values for w_1 and w_2 during their experiment $$\boldsymbol{w}_1 \cdot \sum_{i=1}^{M_t^S} \left(\frac{1}{PC_i - UC_i + e} + \frac{1}{PM_i - UM_i + e} \right) + \boldsymbol{w}_2 \cdot M_t^S$$
[103]	The authors used 1 as a weighting coefficient for *power consumption* and k as a weighting coefficient for *resource wastage* $1 \times \sum_{j=1}^{m} P_j - \boldsymbol{K} \times \sum_{j=1}^{m} UsageEFF_j.$
[9]	The authors used 1 as a weighting coefficient for *execution time* and 1 as a weighting coefficient for *power consumption* as shown below $1 \cdot \theta(EC) + 1 \cdot \delta(ET)$
[59]	The authors used w_1 and w_2 as a weighting coefficient for *power consumption* and *performance degradation* as shown below $\boldsymbol{w}_1 \cdot pow_s(\mathrm{v}_{sum}^s) + \boldsymbol{w}_2 \cdot P(s)$ However, they did not make clear what specific value of w_1 and w_2 they have chosen in their experiment
[105]	The authors used vector values of [1, 1, 1, 1] for weighting vector $w = [w_1, w_2, w_3, w_4]$ as they did not consider any relative importance between different objective functions expressed by below formula $$\boldsymbol{w}_1 \times \frac{F_1 - F_1^*}{F_1^*} + \boldsymbol{w}_2 \times \frac{F_2 - F_2^*}{F_2^*} + \boldsymbol{w}_3 \times \frac{F_3 - F_3^*}{F_3^*} + \boldsymbol{w}_4 \times \frac{F_4 - F_4^*}{F_4^*}$$ where F_1 is *total energy consumption*, F_2 is *Consolidation Fitness* (*for PMs executing two disk-intensive VMs*), F_3 is *Consolidation Fitness* (*for PMs executing a disk-intensive VM and processor intensive VM*) and F_4 is a *limit for processor utilization of a PM.* $F_1^*, F_2^*, F_3^* and F_4^*$ are optimum values for $F_1, F_2, F_3 and F_4$
[106]	For the static scenario, authors used weighting coefficient 1 for *power consumption* and α for *network traffic* as expressed by the below formula: $1.\ Cost_{ser} + \boldsymbol{\alpha}.Cost_{net}$ For the dynamic case, they used 1 as a weighting coefficient for power consumption. For *network traffic* and *VM migration* objectives they used α and β respectively as shown below: $1.\ Cost_{ser} + \boldsymbol{\alpha}.Cost_{net} + \boldsymbol{\beta}.Cost_{mig}$ However, the authors did not make clear what specific values of α and β they have chosen for their experiments.
[107]	The authors used 1 for both objectives of *power consumption by PMs* and *power consumption of communication among VMs* as expressed by the below formula: $1. \sum_{p_j \in P} E(P_j) + 1. \sum_{c \in C} E(c)$
[44]	The authors used α, β, and γ as weighting coefficients for objectives *Resource Usage* (RU), *Server Usage* (SU) and *Bandwidth Usage* (BU) respectively as shown below: $\boldsymbol{\alpha} \cdot RU + \boldsymbol{\beta} \cdot SU + \boldsymbol{\gamma} \cdot BU$ The authors chose $\alpha \ll \beta \ll \gamma$ so that the highest importance is given to minimizing network traffic, the second highest importance is for minimizing server usage and lastly minimizing resource usage has the least relative importance.
[108]	The authors used $\eth_1$, $\eth_2$ and $\eth_3$ as weighting coefficient for objectives *ratio of response time*, *failure rate* and *resource utilization* as denoted by below formula: $\eth_1 \times \frac{tr_j}{tr_{max}} + \eth_2 \times \frac{f_j}{f_{max}} + \eth_3 \times s_j$ The value of $\eth_1$, $\eth_2$ and $\eth_3$ was adjusted according to the cloud user's attention to the three factors

4.2.2 Energy consumption and VM performance

In addition to the single objective algorithm by Kessaci et al. [65] which was previously discussed in Sect. 4.1.2, the authors proposed a bi-objective version of *EMLS-NOC* called *EMLS-NOC-MO* which intends to addresses both energy consumption and performance of VMs. Moreover, among a set of best-found solutions, the priority is given to the solution that packs the highest number of VMs. The VM performance model is defined based on the response time of VMs. The response time is calculated according to a linear relationship with memory increase as shown in Fig. 12 Where $memory_j$ is the memory requirement of *VM j* and mem_usage_i is the current memory usage of *PM i*. To evaluate the *EMLS-NOC-MO*, it was compared to *FFD* and OpenNebula's default Scheduler when it is applied to individual objectives (e.g. energy consumption and performance of VMs). The achieved results from *EMLS-ONC-MO*, are 24% and 9% better than that of OpenNebula's default scheduler when respectively energy and VM performance are objectives. Comparing *EMLS-ONC-MO* to other approaches, the authors also report the superiority of the *EMLS-ONC-MO*. However, no comparison with prominent Pareto-based multi-objective approaches (such as *NSGA-II* [99]) presented.

4.2.3 Resource wastage and power consumption

Gao et al. [95] studied VM Placement as a multi-objective combinatorial optimization problem with two objectives as resource wastage and power consumption. The authors modeled the resource wastage for *j*th PM (W_j) as below:

$$W_j = \frac{\left|L_j^p - L_j^m\right| + \varepsilon}{U_j^p + U_j^m} \tag{20}$$

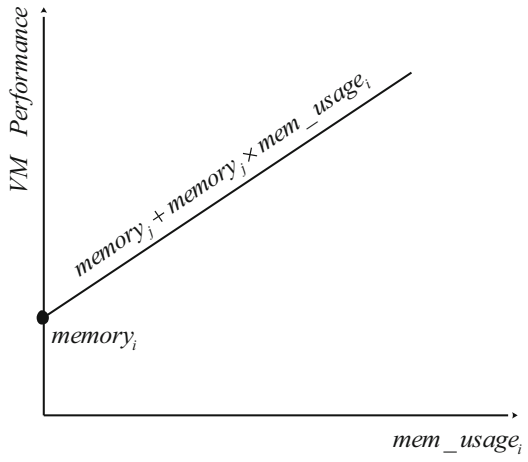

Fig. 12 VM performance model

where U_j^p and U_j^m denote the ratio of used amount of CPU and memory respectively to the total available corresponding resource while L_j^p and L_j^m represent the normalized remaining amount of CPU and memory respectively. ε is a very small positive value.

The presented power consumption model as already discussed in Sect. 4.1.2. is an ant colony optimization algorithm called *VMPACS* which was proposed to simultaneously minimize both problem objectives. The goal is to find a set of non-dominated solutions that provide the best possible trade-off between two objectives with reference to the concept already discussed in Sect. 4.2. The performance of the proposed approach is compared with single objective ant colony (*SACO*) optimization algorithm in [100], a multi-objective genetic algorithm (*MGGA*) that proposed in [81] and a single objective *FFD* heuristic in [49]. Two special performance metrics for the multi-objective algorithms called ONVG [101] and Spacing [102] were employed to evaluate the effectiveness of the proposed algorithm. The conducted experiment demonstrates the superiority of the *VMPACS* to the other algorithms. Furthermore, the experiment result verifies the scalability of the approach in large data centers with many VMs and also shows that *VMPAS* takes less than 3 min to solve a placement problem with up to 2000 VMs. However, the authors did not assess the performance of *VMPACS* in terms of computational complexity when it is compared to *MGGA*, *SACO,* and *FFD*.

In another attempt to minimize the power consumption and resource wastage of cloud data center, Jamali et al. [103] applied an imperialist competitive-based algorithm (ICA); a novel optimization technique which was first introduced by Atashpaz-Gargari and Lucas [104] for tackling real-world solve optimization problem. To reduce the complexity, the bi-objective problem was converted to a single objective one using the weight-based approach. The performance of ICA was compared to the well-known approaches such as Ant Colony, Genetic Algorithm and FFD on a CloudSim simulation environment in terms of power consumption and resource wastage criteria. The simulation results indicate that ICA performs better in reducing the power consumption and resource wastage of PMs as compared to other placement algorithms. However, no comparative analysis of computation time provided.

Zheng et al. [42] proposed a novel solution called *VMPMBBO*. The proposed evolutionary algorithm is a biogeography-based optimization technique and it is intended to find a solution that simultaneously minimizes the resource wastage and power consumption. The authors extended the model in [81, 95] to quantify the cost of the resource wastage along three dimensions of CPU, memory, and bandwidth. The power consumption model is based on

CPU utilization level similar to the model presented in Sect. 4.1.2. Through simulative experiments and using both synthetic and real data, the proposed method was compared with two other multi-objective optimization algorithms: *MGGA* [81] and *VMPACS* [95] and the results show that in most cases *VMPMBBO* has better convergence and also it is computationally more efficient. However, the conducted experimental analysis does not include any performance comparison to other the state-of-the-art evolutionary approaches (such as *NSGA-II* [99]) which has been successfully applied in a variety of optimization problems in different domains. In addition, employing dedicated multi-objective performance metrics seems necessary to evaluate the efficiency of the proposed algorithm precisely.

The problem of VM placement with two objectives of power consumption and resource wastage was also addressed by Gupta, Amgoth [109]. A power consumption model similar to what we presented in Sect. 4.1.2 and a more involved resource wastage model based on CPU demand, memory demand of VMs, maximum memory utilization and maximum CPU utilization of PMs were presented. The main idea behind the proposed methods named as RVMP is the utilization of a new two-dimensional resource usage model (as shown in Fig. 13). The model partitions the CPU and memory utilization space into three different domains according to the degree of balance in resource utilization. Based on this model, the VM migration is limited and this balances the resource utilization. Three different domains in the proposed model are Acceptance Domain (AD): where the residual resource amounts are nearly balanced. That is, there is little resource wastage and it is an ideal case for all PMs. This domain has the highest priority. Balance Domain (BD): where there is no apparent disequilibrium in resource utilization and it is fairly balanced. This domain has the second highest priority. Domain (UD): where there is an obvious disequilibrium in resource utilization. This domain has the least priority. RVMP is divided into two phases as VM placement and VM migration. The decision of placing the VMs are made based on the so-called *Resource Usage Factor* (RUF) which merits the suitability of a PM to host a VM. RUF is calculated based on the resource utilization of VMs and the remaining resources of PMs. VM Migration phase is performed based on RUF and the posterior usage state of PM in the aforementioned resource utilization model. The posterior usage state of a PM with respect to a VM is defined as new usage state of the PM when the VM migrates to that PM and implies the suitability of that PM with subject to the domain in the two-dimensional model which the posterior usage falls into. To evaluate its performance, RVMP was compared with existing algorithms: First Fit, VMPACS [95], MBFD [52] and OBFD [110] in terms of power consumption, resource wastage, overall CPU/memory utilization and the number of active PMs. The simulation results using user-customized VMs and using Amazon EC2 instances demonstrate the superior performance of the proposed algorithm.

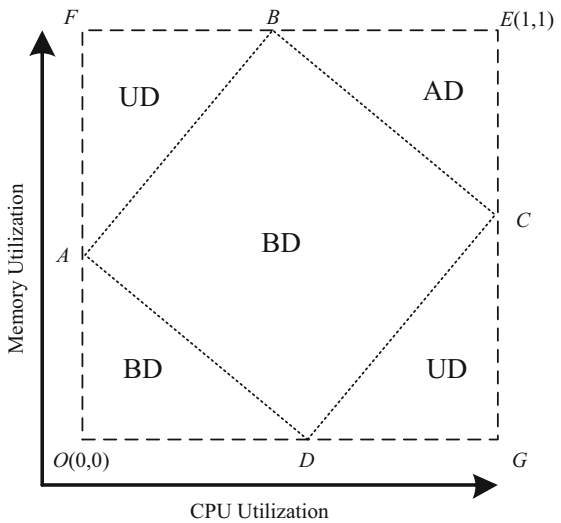

Fig. 13 Multi-dimensional resource usage model

4.2.4 Power consumption and VM execution time

Kansal and Chana [9] proposed the *ERU* (Energy-Aware Resource Utilization) model to efficiently manage the resources in the cloud computing environment. The goal of *ERU* is to reduce the energy consumption of cloud infrastructure without degrading the performance of the user's application which is translated into VM execution time. This model is meant to achieve the maximum possible resource utilization which results in the enhanced energy efficiency of the data center. A weighted sum of two objectives (execution time and power consumption) is minimized. The model for calculating VM execution time (*ET*) is defined as:

$$ET = \sum_{i=1}^{M} ET_i \tag{21}$$

where ET_i is the execution time of VMs running on i^{th} PM and M is the number of PMs. ET_i is defined as:

$$ET_i = \sum_{j=1}^{n} \sum_{k=1}^{l} ET_{ijk} \tag{22}$$

where ET_{ijk} is the execution of k jobs running on jth *VM* on ith PM.

Likewise, the total power consumption (*EC*) is calculated as the sum of power consumption for every single PM (EC_i) as:

$$EC = \sum_{j=1}^{M} EC_i \tag{23}$$

and consequently, the power consumption for every single PM; EC_i; is its power consumption; PC_i; during t units of time as shown below:

$$EC_i = PC_i \times t \tag{24}$$

The key features of the proposed model are: monitoring cloud resources to determine the current level of energy consumption, providing users with requested resources and enhancing resource utilization. As an element of the proposed model, the scheduler module is responsible for finding the best physical nodes for user's jobs. To avoid conflict among different nature of workloads and to prevent potential resource contention, workloads are segregated into CPU-intensive workloads and memory-intensive workloads. The scheduler uses an artificial bee colony (*ABC*) optimization technique to place the dynamic user's workload to an optimal set of physical nodes. Through a simulation-based experiment on CloudSim toolkit [111] the performance of ABC-based technique (called *ERU*) is evaluated against Ant Colony Optimization (*ACO*) [100] and First-Fit Decreasing Heuristic (*FFD*) [112]. The experimental results show *ERU* takes higher time than *FFD* and less time than *ACO* to obtain the final output. In addition, employing the *ERU* approach results in less energy consumption as compared to *FFD* and *ACO* techniques. Specifically, 11% of PMs and 10.7% of power have been saved using ERU over *FFD*. The PMs and power conserved using *ERU* over *ACO* are 6.35% and 6.63% respectively.

4.2.5 Resource fragmentation and number of PMs

Since the resource wastage/fragmentation results from imbalance use of the resource over multiple dimensions (such as CPU, memory and disk space), Li et al. [7] proposed a novel multi-dimensional space partition model to describe the resource usage status of PMs. Figure 14 shows the multi-dimensional space partition model for two different resources. All the resource dimensions are normalized to have capacities in the same range of [0, 1]. The point O indicates that all the resource dimensions are unused and thus the PM is idle. On the other hand, point E refers to the state that all the resource dimensions are exhausted. The model has partitioned into three different domains as (1) acceptance domain (AD): where all the D-dimensional resources are almost finished. A PM with usage state falls in this domain is an ideal candidate for placing a new VM, (2) forbidden domain (FD): this domain implies imbalance in D-dimensional resource utilization

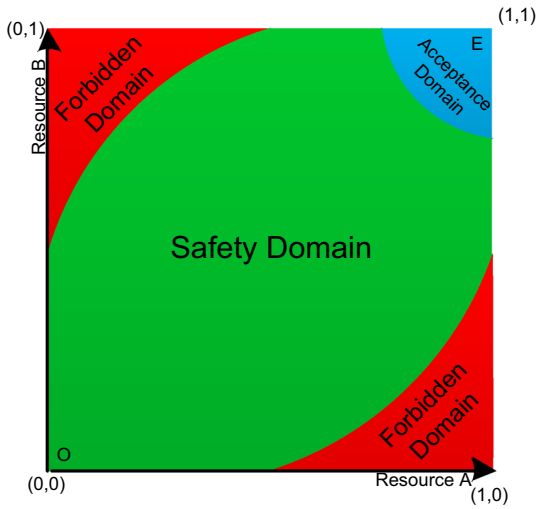

Fig. 14 Multi-dimensional space partition model

and therefore should be avoided. (3) Safety domain (SD): indicates there is no obvious imbalance of resource utilization and considered a balanced case. On top of this underlying model, a dynamic energy efficient VM Placement algorithm called *EAGLE* is proposed to reduce the number of active PMs and therefore decrease the amount of energy that consumed in a data center. *EAGLE* attempts to place the VMs in a more balanced way rather than arbitrary manner. The central idea of *EAGLE* is to achieve a compromise between multi-dimensional resource utilization and minimizing the number of active PMs. *EAGLE* decides to place the VM v on PM p based on a so-called *posterior usage state* of p which refers to the new available resources of p after presumptive placement of v. If posterior usage state of p lies in the acceptance domain, p has the priority to be selected while if its posterior usage state lies in the safety domain it has second priority to be selected. p will not be selected if its posterior usage state lies in the forbidden domain. PMs with identical posterior usage state will be compared according to two other defined metrics $\Re$ and $\mathfrak{D}$. The dynamic feature of *EAGLE* is established on the basis of dividing time into time-slots with equal length Δt. There are $k(\tau)$ VMs to be placed at the τ^{th} time-slot while $N(\tau)$ and $M(\tau)$ represent the total number of PMs and VMs at the τ^{th} time-slot. *EAGLE* starts by initial time-slots (e.g. $\tau = 0$) and records the number of running PMs;$N(\tau)$ at the end of each iteration and repeats the same procedure for the next time slot $(\tau + 1)$. To evaluate the performance of *EAGLE*, it was compared to *FFD* [49] as a well-known heuristic for the bin packing problem. The experiment results for single VM request per time-slot shows that: using *EAGLE* results in 10% less power consumption as compared to *FFD* which is a substantial amount of energy saving for a large data center. Moreover,

for multiple VM requests per time-slot, *FFD* uses 1.15 times of PMs as compared to *EAGLE* which implies that *EAGLE* placement saves 15% of energy cost.

4.2.6 Power consumption and VM performance degradation

Lovász et al. [59] addressed performance degradation incurred when multiple VMs share a single PM in heterogeneous server infrastructure. Running multiple VMs on single hardware is susceptible to resource contention. This is because different VMs send several requests to obtain access to the shared hardware resources such as CPU and this leads to frequent context switching and consequently degrades the performance of VMs. The proposed approach is an energy-aware and performance-aware approach which is meant to make a tradeoff between energy consumption and performance degradation. Besides, authors provided a model to predict the performance degradation overhead (in terms of response time) of a service encapsulated in a VM when it is co-placed with other VMs as compared to the performance of the same service in a non-virtualized environment. The model considers three different parameters which have an influence on the performance degradation of VM v. These parameters are (1) $v\#$: the number of VMs competing for a specific CPU core. (2) s^{CPU}: the total load on CPU core of server s and (3) v^{CPU}: the CPU demand of VM v itself. The mathematical equation below were retrieved to represent the relationship between these parameters and performance degradation of VM v ($p_{virt}(v,s)$) in the virtualized environment when v is placed on server s:

$$p_{virt}(v,s) = p_{no\_virt}(v,s) + v\# \cdot \left(\lambda_1 + \lambda_2 v^{CPU}\right) \tag{25}$$

where $p_{no\_virt}(v,s)$ is the performance of virtual service v on server s in non-virtualized environment and λ_1 and λ_2 are two constants which are experimentally determined.

On the basis of this model, two heuristic algorithms termed *greedy heuristic* and *ModifiedFirstFit* were proposed to approximate the optimal solution for the problem. The average overall performance of the two algorithms is evaluated against four other competitors which are *Load Balancing*, *Maximum density consolidation*, *exhaustive optimal allocation,* and *best from random allocation*. The experimental result demonstrates that the proposed algorithms significantly perform better than other competitors in terms of energy saving. The greedy heuristic provides an additional energy saving of 30%. However, this energy saving is achieved at the price of a higher degree of performance degradation. With regards to computational complexity, both proposed heuristics have the complexity of $O(n.m)$ while the complexity of *best from the random allocation*, the *exhaustive optimal allocation* is $O(n.m)$ and $O(m^n)$ respectively (m number of PMs and n number of VMs).

In another work by Zhao et al. [113], the authors proposed an ant colony-based method named as *PPVMP* to solve the bi-objective VM placement with objectives of power consumption and performance degradation. In dealing with the multiple objectives, authors took advantage of the Pareto concept to find optimal solutions with respect to both objectives simultaneously. *PPVMP* was constructed on the basis of two fundamental power consumption and performance degradation models. The power consumption model denoted by $PW_j(U_j)$ is formulated as:

$$PW_j(U_j) = PW_j^{idle} + PW_j^{dync}(U_j) \tag{26}$$

where PW_j^{idle} is the power consumed by physical machine j when it is in idle state and PW_j^{dync} is the power consumed by physical machine j when it is busy. U_j is CPU utilization level. To characterize the resource contention in PM, three individual performance models for CPU, memory, and network is used. The CPU relative performance of VM running on PM M_j is denoted as mp_c^i and defined as follows:

$$mp_c^i \propto \begin{cases} 1 & \sum_i v_c^i \le M_c^j - M_{c,r}^j \\ \dfrac{M_c^j - M_{c,r}^j}{\gamma_c \sum_i v_c^i} & Otherwise \end{cases} \tag{27}$$

where $M_{c,r}^j$ is reserved CPUs for running M_j. γ_c is the CPU performance degradation parameter. M_c^j is the total CPU for M_j and v_c^i is CPU requirement for VM V_i. Memory relative performance is denoted as mp_m^i and defined as follows:

$$mp_m^i \propto \frac{M_m^j - M_{m,r}^j}{\gamma_m \sum_i v_m^i} \tag{28}$$

where $M_{m,r}^j$ is reserved memory for running M_j, γ_m is memory performance degradation parameter, M_m^j is total memory for M_j and v_m^i is memory requirement for VM V_i. Likewise, the network relative performance is denoted as mp_n^i and defined as follows:

$$mp_n^i \propto \frac{M_n^j}{\gamma_n \sum_i v_n^i} \tag{29}$$

where γ_n is network performance degradation parameter, M_n^j is total network bandwidth for M_j and v_n^i is network bandwidth requirement for VM V_i.

To evaluate the efficiency PPVMP was compared to CMBFD [114], VMPBBO [42] and VMPACS [95] in CloudSim and real OpenStack cloud platform. In terms of both objectives, the authors found PPMP to perform best as

compared to two other methods. According to the authors, the superiority of the proposed approach is mainly attributed to their choice of both power consumption model and performance degradation model. However, as the authors mention, the performance degradation is not considered a target/objective in three compared works and therefore comparison to other VM placement methods that have both objectives in common does make more sense here.

4.2.7 Energy consumption and interference among VMs

Sharifi et al. [105] applied a simulated annealing (*SA*) technique to schedule a number of VMs on a set of PMs in a data center. The goal is to minimize total power consumption in the whole data center while the performance interference among different types of workloads is minimized. The workloads are either processor-intensive workloads or disk- intensive workloads. The scheduler uses a criterion called *consolidation fitness* (*CF*) to merit the consolidation of a set of VMs on a number of PMs before scheduling actually take place. *CF* is calculated by dividing the performance degradation of a set of VMs by the amount of energy saving gained through consolidation as shown by the below equation:

$$CF = PD/SE \tag{30}$$

where *PD* denotes the performance degradation when VMs are consolidated and *SE* is saved energy obtained by the consolidation. Smaller *CF* is, more reasonable is the VM placement. Figure 15 illustrates the way *CF* is calculated. In Fig. 15a the energy consumption (e_1) and the execution time (t_1) are measured for two VMs running on two separate PMs. In Fig. 15b, the same parameters are again measured (denoted by e_2 and t_2) when both VMs are placed on a single PM and the spare PM is switched off. Then *PD* and *SE* are computed by the following formulas:

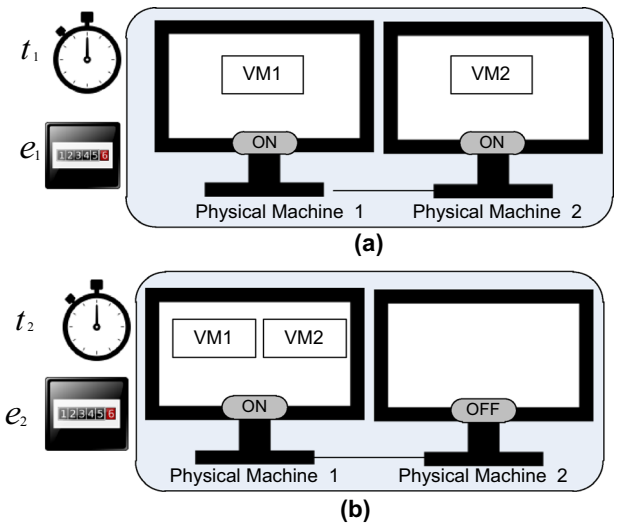

Fig. 15 Calculating CF metric

$$PD = \frac{t_2 - t_1}{t_1} \times 100 \tag{31}$$

$$SE = \frac{e_1 - e_2}{e_1} \times 100 \tag{32}$$

The proposed power model calculates the total power consumption as the sum of power consumed by processor and disk. Firstly, each individual objective is minimized separately using a simulated annealing method to find an optimal point for each objective. Thereafter, a weighted sum technique is employed to transform the original multi-objective problem into a single objective equivalent. Upon optimizing the problem objectives, the proposed method generates a system state as output which includes a binary matrix of mapping VMs to PMs (X_{ij}) along with a binary vector (Z_i) that determines which PM is off or on. This system state can be used by the same proposed method as input to move to the next system state (including X'_{ij} and Z'_i). The difference between the two system states determines which PM should be switched off or on and which VM should be migrated to which PM. However, the authors did not address how frequent the algorithm is executed and under what condition. To evaluate the performance, through a simulative experiment, the presented algorithm was compared to the static algorithms presented in [115, 116] and dynamic load balancing scheduling methods presented in [117]. The comparison was carried out based on the power consumption and computation time of the different algorithms. The results of the experiment indicate that the proposed approach saves 24.9% more energy than two other methods. However, the total execution time of all the VMs for the proposed approach is 1.2% higher than the static method since static algorithms naturally require less time to complete as they have full knowledge of all jobs. The authors also reported the time complexity of the proposed algorithm as $O(M \times N)$ (M number of PMs, N number of VMs).

4.2.8 VMs communication latency and number of PMs

Pascual et al. [39] proposed an evolutionary multi-objective placement policy which attempts to simultaneously minimize the communication latency and a number of active servers for an application. The application is formed by a set of communicating VMs and has to be assigned to any group of physical servers in the data center. The VMs intercommunicate based on a specific layer-based organization similar to what is depicted in Fig. 16. The client is assumed to be aware of the communication need and the interconnection network of his application and hands over

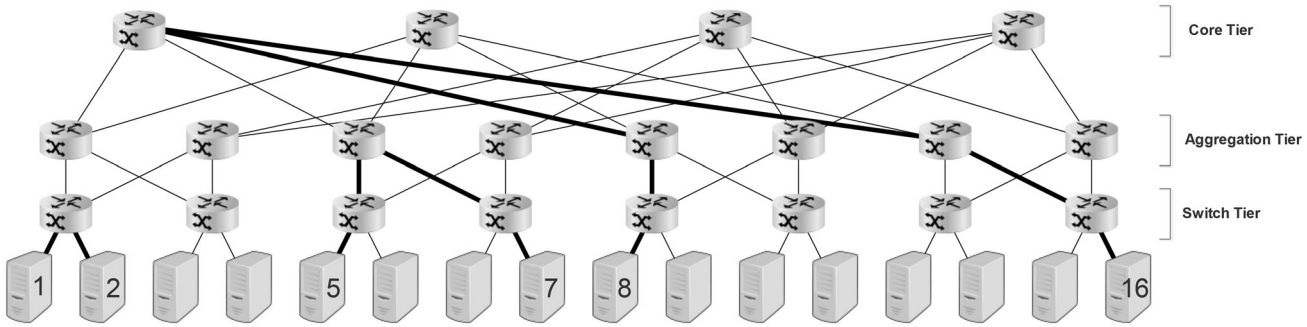

Fig. 16 Representation of physical organization of data center

this information to the provider to be used in the placement process. To reduce the intra-VM communication overhead, the placement policy places the most communicative VMs as close as possible. The proposed model for calculating the communication latency is based on the bandwidth and distance between VMs as shown by the following formula:

$$Latency = \sum_{v_i, v_j \in V} d(v_i, v_j) \times bw(v_i, v_j) \tag{33}$$

where $d(v_i, v_j)$ is network distance between core of PMs assigned to VM i and j, $bw(v_i, v_j)$ is bandwidth required by VM i and j and V is set of all VMs. Distance is measured as a number of hops from sender PM to the receiver PM. For example, as shown in Fig. 16, the distance between the VM assigned to PM 1 and PM 2, or $d(1,2)$ is 2. Similarly, $d(5,7) = 4$ and $d(8,16) = 6$.

Two well-known evolutionary multi-objective algorithms called *SPEA-II* [118] and *NSGA-II* [99] were employed to tackle the placement problem. In addition, the result obtained from common heuristic-based placement strategies: *FF* and *RR* were used as the starting point for the evolutionary optimization technique. The VM placement strategies experimented in a simulation-based environment. According to the results, *SPEA-II* performs better than *NSGA-II* in the majority of cases. Moreover, the results indicate that applying placement policy has a positive impact on both the data center and VMs in terms of VM execution time and energy consumption. Specifically, the average execution time per request is reduced up to 11–19%. Also, for highly loaded data center, the energy saving is between 7.26 and 13.81%. One downside of the proposed approach is requiring clients to specify the application architecture and inter-connection network in advance.

4.2.9 Power consumption, resource wastage, and thermal dissipation

Thermal performance is one of the key indicators in managing a data center [81]. Tightly packed workloads on a small number of servers create hotspot which makes hardware prone to failure and incurs extra cooling expenditure [81]. Designing cooling equipment and ventilation systems are necessary to avoid overheating and performance degradation or even hardware failure [119]. Nevertheless, suitable thermal management policy is still a crucial need to reduce further cooling cost, alleviate hotspots and keep the temperature in a safe range [81]. Xu, Fortes [81] proposed temperature aware placement policies to improve the overall performance by minimizing the temperature of a server alongside other objectives such as power consumption and resource wastage. The proposed policy employs an improved genetic algorithm (called *MGGA*) with a fuzzy multi-objective evaluation to search for a solution which most minimizes the above mentioned conflicting objectives. According to the conducted profiling study, there is a linear relationship between CPU temperature and power consumption as denoted by the following formula:

$$T = PR + T_{amb} \tag{34}$$

where T is a temperature, P is the power consumption, R denotes thermal resistance and T_{amb} is ambient temperature. The presented resource wastage model calculates the wasted resource (W) as a sum of differences between the smallest normalized residual resource (R_k) and others (R_i) as shown in the following formula:

$$W = \sum_{i \neq k} (R_i - R_k) \tag{35}$$

where R_i is the ratio of residual resource to the total resource for resource type i and $R_k = \min_i R_i$. Therefore, the larger the difference among different dimensions is, the more resources are wasted. The result of simulative experiments shows the proposed approach is superior to other approaches such as bin packing algorithm and single objective algorithms (tending to minimize individual objectives) in terms of performance, scalability, and robustness. To validate the performance, the authors showed that, overall, *MGGA* returns lower values for

different objectives. Scalability was evaluated by varying the number of VMs (100–2000) and PMs (50–1000). MGGA takes up to 3 min to solve the problem with 1000 PMs and 2000 VMs. In addition, the execution time for MGGA shows linear growth with a variable number of *S* (population size) and G (number of generations). Finally, to validate robustness, the authors showed that the results of *MGGA* are not sensitive to various values of *S* and *G*.

In addition to the current work, the other study [12] by the same authors address the VM placement in the dynamic scenario and proposes a controller which automatically maps VMs to PMs in order to satisfy the same objectives as the previous study and reduce migration cost.

4.2.10 Power consumption, network traffic, and migration cost

Dong et al. [106] proposed two placement strategies for static and dynamic scenarios. The first algorithm called *VM-P* is a greedy algorithm which was designed for initial placement of VMs. The static problem is abstracted as multi-dimensional bin packing problem with the objective of minimizing a weighted sum of power consumption and network traffic among VMs. Supplementary, *VM-Mig* was developed for the dynamic scenario and it was meant to minimize the weighted sum of the aforementioned objectives coupled with the third objective of migration cost which is defined as a number of migrated VMs. However, the authors did not mention when *VM-Mig* is triggered. Basically, *VM-Mig* uses the same *VM-P* strategy for finding a new placement of VMs on PMs but the outcome of *VM-P* is accepted if the number of migration (as result of the difference between the previous and current placement) is less than a pre-determined threshold. In the event that the new placement requires a number of migrations more than the threshold value, only some of the migrations (less than the threshold) that can improve the performance are accepted. The drawback of *VM-Mig* is poor stability and tendency to get stuck in local optimum.

Through a simulation experiment, the proposed algorithms were compared to *FFD* and *T-opt* and Random algorithm. Overall, the proposed algorithms are found to attain better results in terms of energy consumption and communication traffic. In addition, the time complexity of *VM-P* and *VM-Mig* was reported as $O(m \cdot n^2)$ and $O(nMax \cdot n^2)$ respectively (m number of PMs, n number of VMs, $nMax$ number of loop iterations for finding a placement with a lower number of migrations) which means *VM-P* is computationally more expensive than *FFD* ($O(nlogn)$).

4.2.11 Power consumption by PMs and power consumption for inter VM traffic

As an improvement to their preliminary work [120], Tang and Pan [107] applied a hybrid genetic algorithm (*HGA*), which is a combination of a genetic algorithm and local search technique; for solving VM placement problem. Analogous to the previous work, the objective is to minimize the power consumed by PMs together with the power consumption of a communication network in a data center. The presumed communication network topology is similar to the structure used by Pascual et al. [39]. The power consumption of the communication network is dependent on the number of network equipment such as switches used by VMs to communicate with each other. The communication between pairs of VMs is categorized into four different classes as shown in the example of Fig. 17. These classes are C_1: The communication that does not involve any network device (for two VMs placed on a single PM). The communication between VM 1 and VM 2 in the example of Fig. 17 falls in this class. C_2: The communication that uses only one network device. The communication between VM 1 and VM 3 in the example of Fig. 17 falls in this class. C_3 The communication that involves three network devices. The communication between VM 3 and VM 4 falls in this class. C_4 The communication that uses five network devices. The communication between VM 4 and VM 5 in example falls in this class. The authors approximate the total network power consumption as follows:

$$E(c) = e(c) \times l(c) \tag{36}$$

where $l(c)$ denotes the amount of data needed to be transmitted over communication c and $e(c)$; the power needed to transfer a unit of data within set c. $e(c)$ is defined as follows:

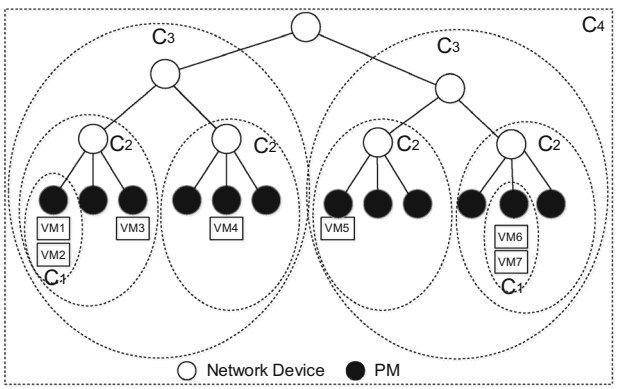

Fig. 17 Different categories of VM communication

$$e(c) = e_i \quad if\ c \in C_i, \quad 2 < i < 4$$
$$e(c) = 0 \quad if\ c \in C_i \tag{37}$$

The result of evaluation demonstrates the superiority of *HGA* over the original genetic algorithm presented in [120] in terms of performance, efficiency, and scalability. In particular, *HGA* was found better in discovering a new solution in the search space, converges faster to the optimal solutions and produce better solutions with respect to minimizing the objective function. Mean total energy consumption of solution found by *HGA* is 27.36–43.90% less than that of original GA while the mean computation time of *HGA* is 73.30–88.61% less than original GA.

Furthermore, by increasing the number of PMs and VMs, *HGA* exhibits nearly linear computation time. However, since the multi-objective problem is transformed into a weighted sum form, determining the proper weight coefficient is not always straightforward and usually needs a time consuming trial-and-error process.

4.2.12 Resource usage, server usage, and bandwidth usage

Kanagavelu et al. [44] developed a greedy approach called *Greedy VM Placement with Two Routing* (*GVMTPR)* to reduce the possibility of network congestion and balance the load in a data center by distributing the traffic in multiple paths. The maximum load on links is considered as a measure of congestion. Besides, the proposed method offers partial traffic protection to enhance link reliability. This is performed by splitting the current traffic flow between two adjacent VMs across two disjoint paths. Therefore, at least one path will be available in the event of a potential single link failure. Specifically, by dividing b units of traffic into b_1 and b_2 units, the minimum of b_1 and b_2 as denoted by *min* (b_1, b_2) is available in the event of single link failure. The partial protection is measured by protection grade which is defined as a fraction of guaranteed bandwidth in case of a single link failure. The protection grade for two paths with b_1 and b_2 units of bandwidth is $\min(b_1, b_2)/b$ where b is a guaranteed unit of bandwidth for a particular flow. In each stage of its greedy procedure, *GVMTPR* attempts to minimize the weighted sum of three costs. These costs are *Resource Usage* (*RU*): as a fraction of resources used in a particular server, *Server Usage* (*SU)* a fraction of active server in the data center and *Bandwidth Usage* (*BU*) as the bandwidth needed for a pair of physical servers; S_x and S_y to communicate with each other weighted by the hop distance $h_{x,y}$. If $B_{x,y}$ is the total bandwidth required for all the VMs placed to S_x to communicate with the VMs in S_y, the *Bandwidth Cost(BU)* for all servers is calculated as:

$$B(U) = \frac{\sum h_{x,y} \cdot B_{x,y}}{B_T} \tag{38}$$

where B_T is the total bandwidth requirement of the traffic.

The performance of *GVMTPR* was compared to the first fit heuristic and random placement and the achieved results demonstrate the effectiveness of the proposed algorithm in terms of bandwidth cost and performance. The authors also report the time complexity of the proposed methods as $O(n^2m^2)$ where n is the number of servers and m is the number of VMs.

4.2.13 Maximum bandwidth occupancy on the uplink of all the tor switches and maximum number of VM partitions of all the requests

Chen et al. [121] proposed Least-Load First Based Placement (LLBP) algorithm to simultaneously minimize the maximum number of VM partition for all the requests and maximum bandwidth occupancy on the uplink of Top of Rack (ToR) switches. The VM placement problem was studied on the basis of the three-layer tree-like architecture of example in Fig. 18. In this topology, ToR switches are connected to the aggregation switches and aggregation switches are connected to the core switches. The accumulated traffic at higher levels links makes them bottleneck and prone to be oversubscribed. Therefore, distributing the traffic evenly across all uplink of ToR switches is necessary to hinder the creation of hotspot. The data center network is assumed to have n ToR switches (as shown in Fig. 18) which are represented by $T = T_1, T_2, \ldots, T_n$. Each ToR switch has the capacity for accepting maximum c VMs at one PM connected to switch. There are $R = \{R_1, R_2, \ldots, R_m\}$ requests from different tenants (each request from one tenant) and each request R_i is for placing set S_i of VMs to more than one ToR switch ($|S_i| > c$). Each

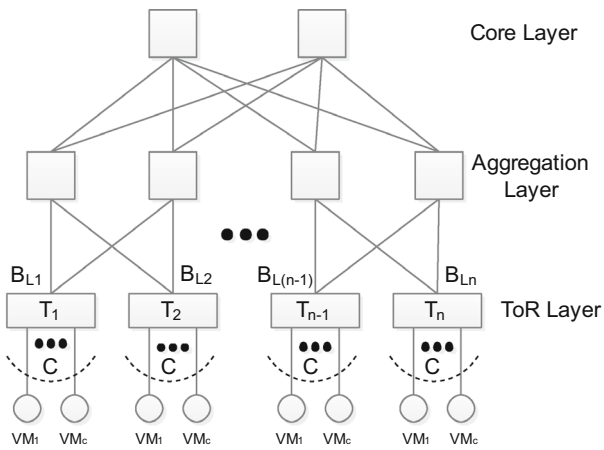

Fig. 18 Three-layers architecture for minimizing maximum bandwidth occupancy of ToR switches

VM under ToR switch contributes a certain amount of traffic toward the higher level. The first objective is defined as:

$$min \max_{k\in[1..n]} B_{Lk} \tag{39}$$

where B_{Lk} is the accumulated bandwidth occupancy on the uplink L_k. To reduce the communication overhead, the VMs of the same request is placed on a few PMs as possible. If T_{R_i} is the subset of ToR switches under which the VMs of request R_i is placed, the second objective is defined as follows:

$$min \max_{i\in[1..m]} |T_{R_i}| \tag{40}$$

where $|T_{R_i}|$ denotes the numbers of switches assigned to VMs of request R_i.

The proposed heuristic algorithm (LLBP) places the requests based on non-increasing order of number of VMs. LLBP tries to place each request in a minimum empty ToR switch that has the capacity for all the VMs of the request. The performance of LLBP was evaluated against Greedy Based Placement (GBP) as a baseline algorithm as well as Longest Processing Time Based Placement (LPTBP) which can generate near-optimal solution but it does not take VM communication locality into account. According to the simulation results and based on the minimum and maximum recorded values of bandwidth occupancy of the uplink of all ToR switches, LPTBP performs best while LLBP is in the middle and GBP is the worst. The authors also observed that GBP takes full advantage of communication locality property (as reflected by the second objective), LPTBP is able to equally spread the traffic across all the uplinks of ToR switches (as reflected by the first objective) and LLBP is effective in simultaneously balancing both objectives.

4.2.14 Response time, failure rate and resource utilization

Chen and Jiang [108] developed a fault-tolerant VM placement method to guarantee the reliability of cloud applications. The proposed method considers three factors as constraints, namely: *response time*, *failure rate* and *resource consumption* for a cloud application running on a VM. In addition, four well-known fault tolerant strategies as *Retry*, *Recovery Block*, *N-Version Programming* and *Active* are employed. The objective function is defined as minimizing the weighted sum of the *ratio of response time*, the *ratio of failure rate* and the *resource utilization* for a cloud application running on a VM. A two-phase VM placement algorithm proposed. In the first phase, the best objective function value for each fault-tolerant strategy is obtained. In the second phase, the VM placement is solved based on the result from the first phase. The authors compared the performance of the proposed fault-tolerant approach to three other fault-tolerant strategies (*NOFTPlace*, *RandomFTPlace*, *ResourceFTPlace*) with the constant increase of constraint and the result shows the achieved value of the objective function for the proposed approach is less than that of other approaches. The authors also reported the time complexity of the proposed VM placement algorithm as $O(a \times v \times n)$ where v is the number of VMs, n is the number of PMs and a is the number of fault-tolerant strategies.

4.3 High-performance computing (HPC) applications

Cloud computing can be envisioned as a potentially cost-effective solution for high-performance computing applications. This can be particularly attractive for users with small computing capability who are unable to establish their own cluster infrastructure. The pool of interconnected commodity computers, as well as virtualization technology, makes the cloud a considerable choice for HPC applications [122]. However, the difference between the nature of HPC application and current cloud architecture might hinder effective utilization of cloud infrastructure for HPC purposes. An HPC-aware VM placement strategy is expected to improve the performance of HPC applications because in HPC with loosely coupled architecture, internal computing nodes frequently interact with each other. Some works such as [122, 123] studied a VM placement strategy with taking into account the characteristics of HPC in the cloud.

Gupta et al. [122] explored the challenges and advantages of HPC oriented VM Placement technique for the cloud computing environment. Two techniques for optimizing the VM placement with subject to HPC applications are implemented. These techniques are *topology awareness* and *hardware awareness*. Topology awareness requires providing the knowledge of network topology to the HPC application. Typically, in the context of cloud, the cluster topology is transparent to the users. However, for an HPC application, the goal is to place VMs into PMs having the least possible distance from each other in order to reduce communication overhead. To address this issue, the authors attempted to place all requested VMs on the same rack rather than randomly distribute them over the data center. Hardware awareness requires providing the specification of the underlying hardware to the HPC application. HPC applications are composed of a number of iterations. There are two different phases to be performed in each iteration as computation and communication/synchronization. The next iteration cannot be started unless all other processing nodes have fully completed their previous iteration. Cloud infrastructures consist of heterogeneous commodity PMs.

When there is a slow PM that it takes a longer time to complete iteration, the time in faster PM wasted and this degrades the overall application performance. In compliance with hardware-aware characteristic, a proper VM placement strategy is expected to place all VMs to a set of PMs with equal computing power. Authors address this issue by attempting to place all the requested VMs on the identical type of processor. The topology awareness and hardware awareness are implemented on top of OpenStack scheduler layer [124]. The OpenStack Scheduler is responsible for receiving the VM request and determining the proper PM for hosting the VM. An evaluation was conducted based on *OpenCirrus* [125] test-bed. The result indicates that using a topology-aware mechanism results in a 5% improvement in performance as compared to the random scheduling. In addition, after applying the hardware-aware technique, *20% of time * N CPU-Hours* (N: number of processors used) improvement is achieved in terms of execution time.

Due to the communicative nature of HPC applications, they are often subject to competitive access to *Shared Last Level Cache (SLLC)*. This incurs a serious issue called *cache contention*. Cache contention overshadows the performance isolation offered by virtualization to HPC applications running within VMs. Jin et al. [126] addressed the performance degradation resulted from cache contention of applications in *HPC* cloud. An enhanced reuse distance analysis with accelerated cyclic compression algorithm is employed to classify the *HPC* applications based on their cache access behavior. According to this classification, the *HPC* cloud applications are divided into three different categories as *Cache Pollution Applications*: which occupy a large amount of cache capacity, *Cache Sensitive Applications*: which strongly depends on the available cache resources, and *Cache friendly Applications*: which consumes a small amount of cache capacity. In addition to the reuse distance analysis, *CCAP: Cache Contention-Aware Virtual Machine Placement* method is designed to cope with the cache contention problem. *CCAP* dispatches VMs to the distinct cores based on the applications' cache behavior information. Indeed, *CCAP* tends to minimize interference of cache sensitive applications and cache pollution applications and thus mitigates the negative impact of cache contention. The result of the evaluation shows that *CCAP* significantly enhances the performance of cache sensitive applications when they are co-scheduled with cache pollution applications.

Similar to [126], Kim et al. [127] addressed the performance degradation of the applications hosted in multiple VMs. The VMs are to be mapped to PMs with the modern multi-core processor architecture. In this architecture, each individual core has its own private cache while *last-level cache (LLC)* and memory bus are shared among different cores. Co-located VMs on a PM with multi-core processor contend for accessing *LLC* and memory bus and therefore performance degradation arises due to interference among applications. A performance model is proposed based on two measures: *Interference Intensity* and *Interference sensitivity*. Interference intensity is a measure of how much an application hurts other co-located applications and interference sensitivity is a measure of how an application suffers from other co-located applications. Based on this performance model, a VM placement algorithm called *swim* is presented. *Swim* aims to minimize the average performance degradation ratio of all the applications. The main idea behind *swim* is to co-locate high interference-intensive VMs with less interference sensitive VMs. The experimental results show that applying *swim* causes similar performance degradation as compared to the optimal allocation.

In another study, based on performance analysis, Mc Evoy et al. [128] conclude that chosen strategy for mapping virtual clusters to the physical resource together with the inter-communication pattern between the application processes has a significant impact on the performance of *HPC* parallel application.

5 Taxonomy

This section provides a thematic taxonomy on VM placement approaches as depicted in Fig. 19. The presented taxonomy is organized based on the several parameters and aspects such as uniformity of PMs/VMS, number of clouds, operation mode, problem objectives, methodology, number of objectives and resource demand mode. On the basis of this taxonomy and after discussion of the aforementioned aspects in the following sections a detailed comparison of different methods is also presented in Table 4.

5.1 Uniformity

VM placement is defined in two different contexts, namely: VM placement in the heterogeneous environment data center and homogeneous environment. In the heterogeneous platform, PMs have different hardware specifications (such as CPU speed, memory size, and disk storage amount) [129] while in the homogeneous platform all the machines have an identical hardware configuration. In a cloud computing environment where old computing nodes and new ones operate alongside, the placement is often carried out on a heterogeneous platform. As a result, an assumption of homogenous PMs for a placement strategy is less realistic and can be a limiting factor in practice. In the design and implementation of a placement policy, the uniformity mode of underlying hardware should be taken

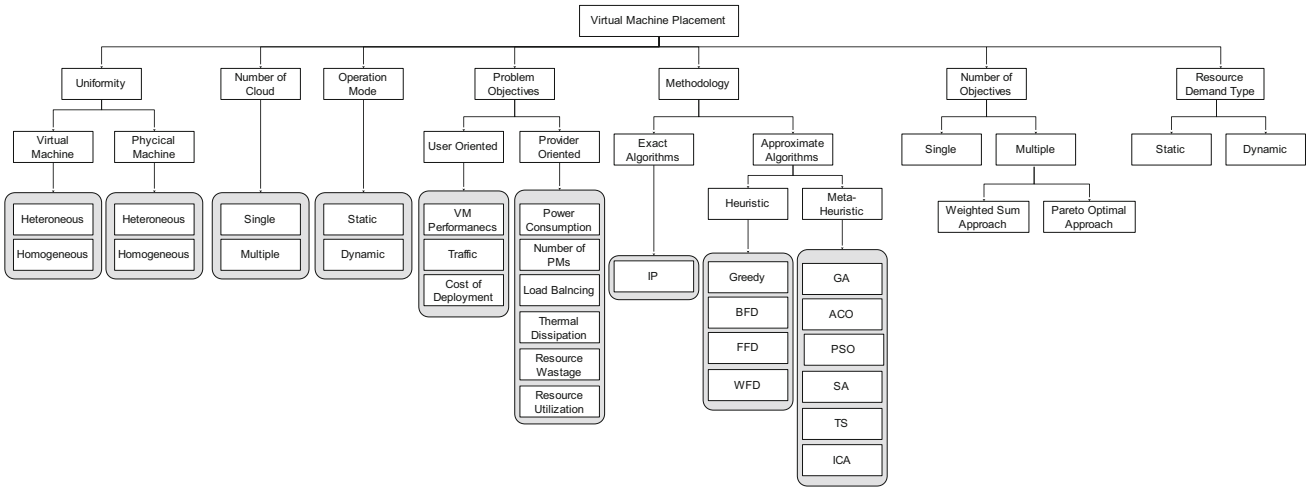

Fig. 19 Taxonomy of the state-of-the-art VM placement techniques

into account. Aside from uniformity of PMs, the uniformity of VMs refers to mapping a set of VMs with identical hardware specification onto a set of PMs.

5.2 Number of clouds

Although the majority of works studied the VM placement problem in a single cloud environment, the problem is also addressed [40, 90, 130, 131] in the multi-cloud scenario. Figure 20 shows the architecture for multi-cloud scenario. In a multi-cloud scenario, a cloud client requests for a collection of VMs to be deployed across multiple cloud providers. Each cloud provider offers different pricing plan, diverse VM instance types and has different resource management interfaces [132], Due to complicated decision-making task in such a scenario for a typical client, often, a cloud brokering mechanism is required to serve as an intermediary between client and providers [40, 132]. The broker middleware gathers information from different individual clouds and then optimally distributes the VMs to the most suitable servers across multiple cloud systems based on the VMs requirements and clouds resources specifications [132]. In addition, the cloud broker provides a uniform management interface to the client with a transparent view of a heterogeneous set of providers regardless of particular cloud provider technology [40, 90]. The advantage of a multi-cloud service for the client is reducing cost, fault tolerance and enhancing service reliability. In dynamic placement modes, one of the salient challenges is communication overhead due to VM migration between different cloud providers [132]. Again, in case of tightly coupled VMs with large inter-VM traffic, the multi-cloud scenario results in high communication overhead between different clouds [90]. Furthermore, in long term period, the cloud specification such as prices schemes, VM instances types is subject to frequent revision and therefore the placement algorithm is required to be recurrently running to adapt the resource allocation with latest changes in cloud provider [40, 90].

5.3 Operation mode

The problem of placing a set of VMs on proper PMs is defined under two different modes: *Static* (*or Initial or offline*) placement and *Dynamic* (*or online*) placement. The static placement is to place a number of VMs at once on an unloaded data center with subject to VM requirements and resources capacities of PMs. The static placement is also often performed when the system resumes operating after a period of idleness or reset. The decision on a static placement plays an important role in the overall data center performance since the large change in initial VM assignment incurs extra migration and therefore large communication overhead is imposed. In addition, the static placement is performed less frequently compared to the dynamic placement and has long term effect since extensive changes incur large overhead [12]. On the other hand, in dynamic placement, VMs are re-assigned to PMs due to the unforeseen changes in VM requirement, VM termination, halt and launch of new VMs. In a dynamic placement scenario, an instant decision at run-time should be made to minimize the migration overhead and improve the overall system performance [41]. Indeed, for each mode of placement different strategy is sought.

5.4 Objectives

In general, the objectives for VM Placement can be in two different types as Cloud Service Provider (CSP) oriented or user-oriented. The *CSP* oriented objectives are defined to

Table 4 Comparison of different virtual machine placement schemes

Methodology	Scheme	Heterogeneous PMs	Mode of operation	Number of clouds	SLA aware	Traffic aware	Resource dimensions
• First fit decreasing	[62]	✔	Dynamic	Single	✔	×	C
	[47]	×	Dynamic	Single	✔	×	M
• Best fit decreasing	[86]	✔	Static	Multiple	×	×	CMDB
	[52]	✔	Both	Single	×	×	C
	[67] [a]	×	Both	Single	×	×	CMB
	[64]	✔	Dynamic	Single	✔	×	NA
• Other greedy and heusristics	[44]	×	Static	Single	×	✔	CMB
	[59]	✔	Dynamic	Single	×	×	CMDB
	[79]	NA	Static	Single	×	×	NA
	[106]	×	Dynamic	Single	×	✔	CMD
	[121]	NA	Static	Single	×	✔	NA
	[63]	✔	Dynamic	Single	✔	×	C
• Linear Programming and	[67]	×	Both	Single	×	×	CMB
• Integer Programming	[11]	NA	Static	Single	×	×	NA
	[90]	✔	Static	Multiple	✔	×	NA
	[40]	✔	Dynamic	Multiple	✔	×	CMD
• Genetic algorithm	[87]	×	Dynamic	Multiple	×	×	CMBD
	[65]	✔	Static	Multiple	✔	×	NA
	[83]	×	Dynamic	Single	×	×	CM
	[107]	×	Static	Single	×	×	CM
	[39]	NA	Static	Single	×	✔	NA
	[81]	×	Static	Single	×	×	CM
• Ant colony optimization (ACO)	[95]	×	Static	Single	×	×	CM
	[69]	×	Static	Single	×	×	CM
	[48]	Both	Static	Single	×	×	CM
	[74]	Multi-core	Static	Single	×	×	NA
	[113]	×	Static	Single	×	×	CMB
	[68]	✔	Dynamic	Single	×	×	CM
• ACO + PSO	[82]	×	Dynamic	Single	×	×	CMD
• Convex optimization theory	[80]	NA	Static	Single	×	✔	CMD
• Simulated annealing	[105]	×	Dynamic	Multiple	×	×	CM
• Artificial bee optimization	[9]	×	Static	Single	×	×	CM
• Tabu search	[73]	NA	Static	Single	×	✔	NA
• EAGLE [7]	[7]	×	Dynamic	Single	×	×	CM**
• Biogeography-based optimization	[42]	✔	Static	Single	×	×	CMB
• Imperialist competitive algorithm	[103]	×	Static	Single	×	×	CM
• A two-phase algorithm	[108]	×	Static	Single	×	×	CM
• RVMP	[109]	✔	Dynamic	Single	×	×	CM
• Particle swam optimization (PSO)	[51]	×	Static	Single	×	×	CM
• Memetic algorithm	[71]	×	Dynamic	Single	×	×	C

C CPU, *M* memory, *D* disk, *B* bandwidth, *NA* not available

[a]The authors also used FFD, BFD, WFD and AWFD algorithms

[b]The authors also did other simulation with three types of resources (D = 3) but did not mention the name of resources

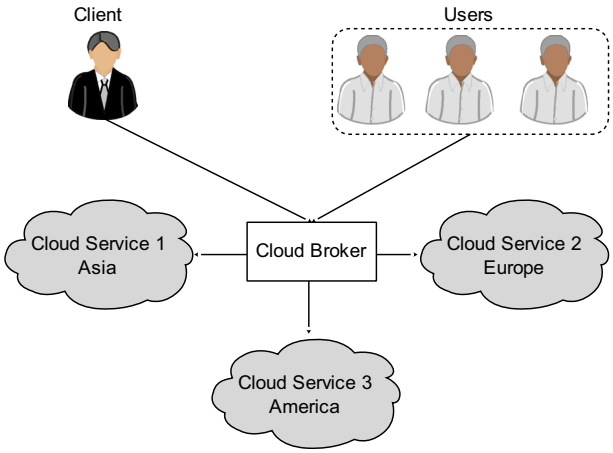

Fig. 20 Multi-cloud architecture

fulfill the requirement or minimize a cost for the sake of data center provider's benefit while the user-oriented objectives are meant to serve users in a faster and affordable way. For example, power consumption as a common objective is meant to reduce the operational cost for a data center provider. In contrast, the cost of deployment is a typical objective of interest for a cloud user.

5.5 Methodology

In this section, a spectrum of the most prominent algorithms proposed in the literature for dealing with VM placement is briefly described. Finally, a comparison of these algorithms based on their advantage and limitation is presented in Table 5. The algorithms proposed for solving the VM placement problem can be generally classified into two different categories as *exact algorithms* and *approximate algorithms*. The exact algorithms are guaranteed to provide an optimal solution to the problem. However, they are not practical because of their high computational time unless they are used to solve small-sized problems. If finding an optimal solution takes a long time, then we should make a tradeoff between optimality and efficiency. In practice and for a large-sized problem having a large number of VMs and PMs as input (in case of VM placement problem), approximate algorithms (either heuristics or meta-heuristics) are utilized to deliver a sub-optimal solution within a reasonable amount of time. The heuristic algorithms are specific problem-dependent methods which take advantage of the problem specification to solve the problem. On the other hand, meta-heuristic algorithms are generic problem-independent methods which can be used to solve a wide range of complex problems.

5.5.1 Greedy heuristics

A greedy algorithm is a simple optimization algorithm which goes through a series of stages. In each stage of the algorithm, the best possible choice should be made for that particular stage. A greedy algorithm makes a sequence of local optimal choices which are anticipated to construct a globally optimal solution at the end [133, 134]. Even though, the optimal solution is not always guaranteed in these techniques, a reasonable time; especially for large problem size; is spent to achieve a sub-optimal solution in contrast to exact approaches [67].

The VM Placement is often formulated as a variant of multidimensional bin packing problem. In multi-dimensional bin packing problem, a set of objects with different dimensions are to be packed into a number of bins with multiple dimensions. The goal is to pack the objects into a minimum number of bins. The bin packing problem is also recognized as an NP-hard problem. That is, there is no optimal solution with polynomial time complexity. When VM Placement is reduced as bin packing problem, PMs represent bins while individual VMs stand for objects. By far, a number of well-known heuristics have been proposed to deal with the problem [49, 67, 135]. Most of these heuristics are greedy-based algorithms. In the following, a few well-known greedy-based heuristics which are intended to address VM Placement as a variant of multidimensional bin packing problem are described.

First Fit Decreasing (FFD) FFD is one of the well-known heuristics for tackling multidimensional bin packing problem. The basic idea of FFD is to sort the list of VMs based on the descending order of certain criteria first and then place VMs sequentially from VM with largest criteria to smallest one to the first PM with sufficient residual resource. The criteria can be certain resources such as (CPU, memory) [67] or a single scalar which is calculated as a function of individual resources in a VM. FFD is proven to allocate VMs to not more than $\frac{11}{19}OPT + 1$ PMs where OPT is the optimal number of PMs [112].

Best Fit Decreasing (BFD): BFD first makes a list of VMs which is sorted according to the descending order of certain criteria similar to FFD. However, in the second step, VMs are sequentially placed on a PM with least sufficient residual resource. BFD is also shown to require $\frac{11}{19}OPT + 1$ PMs in the worst case.

Worst Fit Decreasing (WFD): In WFD heuristic, the list of VMs is first sorted according to certain criteria like two other aforementioned heuristics. However, in the next step, VMs are sequentially placed on a PM with largest sufficient residual resource.

Table 5 Comparison of existing solutions for solving VM placement problem

Algorithm	Advantage(s)	Limitation(s)
• Greedy heuristic	• Efficiency (Quickness) • Simplicity • Can provide a good approximation of the optimal solution	• May not always return the globally optimal solution
• Integer programming model	• Versatility • Simplicity	• Only one objective can be maximized/minimized • Many real-world problems cannot be expressed in a linear form • The high computational effort required for large problems • It is applicable to the static situation only
• Genetic Algorithm	• Can be parallelized with little effort due to inherent parallelism • Can escape from local optima	• The slow rate of convergence to desirable solutions and high computational effort • The performance of algorithms is highly sensitive to the chosen parameters • Premature convergence
• Ant colony optimization	• Can avoid premature convergence thorough distributed computation • Rapid solution finding as a result of positive feedback	• Slow convergence • Poor performance in solving the problem with large space • Stagnation
• Simulated annealing	• Robust and easy to implement	• A large number of objective evaluation make it slow especially when the objective function is computationally expensive
• Artificial bee colony optimization	• Simplicity • High flexibility • Robustness • Fewer control parameters	• Slow convergence rate • Prematurely falling into local optima
• Tabu search	• The convergence speed is dependent on the initial solution • Is not suitable for continuous search spaces	• Utilization of memory to guide the search beyond local optimality
• Imperialist competitive algorithm	• Inherent parallel mechanism • Applicable to a wide range of optimization problems • Simplicity • Scalability	• Low convergence speed • Susceptible to premature convergence • Requires parameter tuning
• Memetic algorithms	• Accelerated search • Higher chance of convergence	• As it inherits GA principle, its performance can be still sensitive to the choice of initial parameters

5.5.2 Linear programming and integer programming

Linear programming (*LP*) is a mathematical technique for optimization in which the objective functions to be optimized (maximized or minimized) is represented in a linear form. Besides, a number of linear equality and inequality constraints should be satisfied. In particular, integer programming is a special form of linear programming in which variables can take integer values only [136]. A variety of real-world problems can be modeled and solved by *LP*. Generally, an *LP* is expressed as follows [137]:

$$\begin{aligned} &Maximize/Minimize\ Cx \\ &Subject\ to: \quad Ax \leq B \\ &and \qquad\quad x \geq 0 \end{aligned} \tag{41}$$

where C is a vector of constants, $\boldsymbol{x}$ is a matrix of variables and $\boldsymbol{A}$ and $\boldsymbol{B}$ are matrixes of coefficients. After representing a problem in LP form, a specific solver such as CPLEX [92] is used to solve the problem.

5.5.3 Genetic algorithm

Genetic algorithm (*GA*) which was first introduced by Holland [138] is a search technique for solving the optimization problems. *GA* emerged as a popular and powerful

approach in finding near-optimal solutions for the complex problem with large search space in different domains. The technique is an inspiration of biological evolution that takes place in nature as expressed by Darwinian Theory of *natural selection*. In the natural environment, the fittest living organisms are more likely to resist against diseases and other dangers and eventually are able to survive and reproduce the next generation which have even fitter individuals than the previous generation. A simple genetic algorithm (*SGA*) simulates the natural evolution through a series of computer instructions. *SGA* commences with an initial population of random individuals. Each individual represents a candidate solution to the problem at hand. The fitness of an individual commensurate to the degree it minimizes/maximizes the problem's objective function. To create a new generation of individuals, first, individuals with the highest fitness value are chosen through a particular selection mechanism. Then, crossover and mutation operators are applied to the selected parent in order to produce new offspring. The iterative evolution from one generation to the next is continued until a solution with satisfactory fitness is discovered.

5.5.4 Ant colony optimization

Ant colony optimization (ACO) is population-based metaheuristic which is inspired by the foraging behavior of ants in nature [139]. This behavior enables real ants to find the shortest path from their nest to the food resources [140]. This is carried out by depositing pheromone trail on the ground as a medium of intercommunication among ants. This characteristic is simulated by artificial ants to solve combinatorial optimization problems [141].

5.5.5 Simulated annealing

Simulated annealing (*SA*) is a popular search heuristic for combinatorial optimization inspired by the annealing process in metallurgy where a metal is heated to its melting point and then it slowly cooled again [142]. In each iteration, *SA* generates all the next moves at the neighborhood of the current solution. Then a move is randomly picked. The moves that improve the quality of the solutions are always accepted while non-improving moves are accepted with a certain decreasing probability of less than one [143]. In fact, the key feature of SA is to overcome getting stuck in local optima which is occurred in older techniques like hill-climbing [144].

5.5.6 Artificial bee colony optimization

Artificial bee colony (*ABC*) which was first introduced by D.Karaboga in [145] is a subclass of swarm-intelligence based algorithms that imitate the collective intelligence of honeybee swarms to solve various optimization problems. In *ABC* algorithm the colony of artificial bees split into three (3) groups, namely: *employed* bees that forage for food, *onlooker* bees who observe other bees and *scout* bees that randomly search for new food sources. The position of the food resource represents a potential solution to the problem. For each food source, only one employed bee is designated and the quality of the solution is proportional to the available amount of nectar in the source. The employed bees forage for food sources and when they bring the nectar to the hive they share the gathered information of food source, its position and its quality with onlooker bees through a so-called waggle dance as a medium of communication. The onlooker evaluates the information from employer bees and chooses the best food source to forage [146]. While employer bees exploit the search space by slightly modifying the food source's position with the hope of improving the solution, scout bees perform exploration by randomly discovering new promising sources.

5.5.7 Tabu search

Tabu search (*TS*) is a local search strategy which is developed by Glover [147] to cope with trapping into local optima in *SA*. The strategy is characterized by its capability of memorizing a history of previously encountered solutions. *TS* uses a short term memory called *Tabu list* to record the recently explored solutions and therefore avoids re-visiting the solutions. This is carried out to prevent recycling problem. A tabu search algorithm begins by evaluating all the neighbor solutions to the current solution. Then, a solution with the highest quality is selected and the tabu list is updated accordingly.

5.5.8 Imperialist competitive algorithm

The imperialist competitive algorithm (ICA) was first introduced in [104] to solve the real-world optimization problems. The algorithm mimics the imperialist competition among empires. Analogous to Genetic Algorithm, ICA commences with an initial population of random individuals. Each individual is called a *country*. A country can be either *imperialist* or *colony*. During the imperialist competition process, the powerful imperialist which represent a better solution to the problem makes the weaker ones collapse and take control of their colonies. The same competition will be iterated until a single empire including an imperialist along with its colonies is left at the end. The final imperialist represents the best-found solution to the problem.

5.5.9 Memetic algorithm

Memetic algorithm (MA) was first introduced by Moscato [148]. MA is population-based metaheuristics and a hybrid form of genetic algorithms (GA) and local search techniques [149–151]. Incorporating the local search capability into the genetic algorithm accelerates its search and increase the chance of convergence [151]. Similar to GA, MA begins with a population of random members. Then, a local search is applied to each member to improve the quality of the solution it represents. Thereafter, new offspring are produced by performing the crossover and mutation operators. Again the local search is applied to new offspring forming the new population. Producing new generations are continued until convergence occurs [151, 152].

5.6 Resource demand type

Most of the studies on VM Placement assume the VM resource demands are constant value over time. This type of demand is called static/deterministic demands [153]. Static demands are simply compared with the residual capacity of target PM at the time of placement. Unlike VMs with static demand, VM with dynamic demand changes their amount of resource requirement during their lifetime. According to some recent studies [154–156], the VM's demands for particular resources such as network bandwidth can be fluctuating and therefore difficult to anticipate at initial [157]. In this case, mean or maximum VM demand is used as an estimated value although the estimation is not always accurate and may result in over-provisioning or resource wastage [153, 158]. Some works like [153, 158] investigated VMs with stochastic demands and used a probabilistic model based on a random variable to represent the uncertainty of future demands [157, 158]. In one of the attempts to estimate dynamic demands, Isci et al. [159] introduced a resource demand estimation technique which is meant to be lightweight, accurate and general. Through an experiment on synthetic and real data, authors found the technique is able to significantly improve the efficiency of dynamic VM placement.

6 Open issues and future directions

This section is to highlight a number of salient research directions which are less focused in the past and thus deserve more attention by researchers. The presented items suggest a potential platform for future research work in the domain of VM placement.

6.1 Thermal-aware placement policy

A significant portion of electricity spent on computer equipment is transformed into the heat. Operating in high temperature reduces the lifetime of hardware parts and makes them less reliable and susceptible to failure and malfunctioning. Therefore, keeping hardware items in a safe temperature zone is always necessary. Even though, today's the cooling systems are widely exploited in modern data centers, using these systems is subject to additional expenditure for purchase, maintenance and electricity consumption. One of the potential ways for minimizing the heat dissipation is through continuous monitoring of the thermal state of PMs and re-placement of VMs once a hotspot is created [160]. As a result, the relieved PM requires less cooling power. Further research work is required to address the thermal topology and analysis for a data center in order to facilitate the efficient placement of VMs.

6.2 Price aware placement policy in a multi-cloud scenario

Today, the diverse number of cloud providers established a competitive market for users. Each cloud provider offers diverse service plans with different time-varying price schemes (dynamic or static), specifications and value-added features. This circumstance offers an opportunity for the users to select the most affordable service with a maximum level of the desired quality. To best of our knowledge, there are few works [90] to address this issue and further research seems necessary to study, design and implement schemes for price aware placement techniques in the multi-cloud scenario.

6.3 Security

Although VM Placement has been studied in several performances oriented aspects, the problem is less explored from the security and privacy perspective. For instance, the client might require two specific VMs not to be run on the same PM and this is to prevent potential leakage of business secrets to competitors. In addition, clients are might be interested to restrict the placement of VMs to some data centers with certain geographical points (e.g. due to some legal issues.)

6.4 Scalability

For many current VM placement approaches, there is no much rigorous analysis and evaluation to demonstrate the proposed strategy efficiently scales up to the modern

gigantic data centers with thousands of VMs and PMs. Thereby, a potential future research direction could be studying the extension of current multi-cloud placement techniques to cope with real-world large scale data centers.

6.5 A dynamic placement scheme from the scratch

Although dynamic VM placement was studied in a number of research works, majority of these works are designed on the basis of a recurring static placement across different time-slots or consolidation states or they have merely focused on variable resource demands of VMs [7, 67, 83]. In these works, at the beginning of each time-slot, a static VM placement algorithm is invoked again and the difference between two outputs/states (for previous and current time-slot) is calculated. The difference determines the VMs that should be migrated to other PMs as well as the PMs that should be powered-on/off. However, to best our knowledge, none of these works addressed the VM placement as a native dynamic scenario in which every VM has its own lifespan and during its lifetime it may undergo load change. A potential future direction in this context is the investigation of a comprehensive dynamic mechanism for VM placement that addresses dynamic creation, dynamic deletion, dynamic resource demand of VMs, failure of PMs and addition of new PMs.

6.6 Renewable energy resources

Modern data centers have begun moving toward exploiting renewable green energy resources such as solar, wind and tidal power as a replacement to the energy that is supplied from the electrical grid. In a geographically distributed cloud system having a number of data centers sites spread over different points of the globe, each data center may be operated using a particular type of renewable energy. Availability of these energies is subject to time or weather condition in their location. The efforts are necessary to make efficient use of these energies. Achieving this requires design and implementation of VM placement strategy which works in accordance with the availability of energy resources. For instance, during night time, PMs in some of data centers around the globe lose their source of solar energy and therefore their running VM should be migrated to other PMs in data centers currently operating in the daytime. Initial placement of VMs also must consider the availability of energy in each data center location before placing the VM.

6.7 Multi-core processors

To best of our knowledge, so far a little research works have been carried out to study, design and development of the VM placement policy that is compliant with modern multi-core processing architecture. In particular, an elaborated analytical power model seems essential to precisely estimate the degree of energy consumption in these systems. In addition, an adequate effort should be devoted to mitigating the potential performance degradation due to the contention of a shared resource in these systems.

6.8 Resource dimensions

Majority of research works discussed in this paper addressed the VM placement problem with main focus on CPU and memory as two prime resource types as reflected in Table 4. However, in the era of modern applications such as online gaming, video streaming and augmented reality, other resource dimensions such as Graphical Processing Unit (GPU) as bottleneck can also be important factors in resource management and significantly contribute to the amount of power consumption in a data center. Therefore, prospective research in this domain should devote adequate effort to study the impact of GPU in the designed power model and resource management policy.

7 Conclusion

One of the most critical issues for large scale data centers is the substantial growth of power consumption. Efficient management of hardware resources can significantly reduce the amount of power consumption in a data center. Many of the servers in a data center operate at low utilization level. Minimizing underutilized servers to an optimal number of fully utilized servers and turning off the spare servers would significantly help in cutting down the rampant electricity consumption. In a virtualized data center, VM placement is a primary and complex decision which can affect the overall energy consumption in a data center.

In this paper, we have investigated several proposed methods in the literature for dealing with VM placement problem. The problem, as we already saw, is defined under diverse settings in terms of number of objectives (either single objective or multi-objective), type of objectives (energy consumption, resource utilization, number of PMs and etc.) and presence of constraints. These settings vary from one scheme to another. Based on our observation from different research works, it appeared that the choice

of problem setting is highly dependent on the particular context and priorities. For example, when energy efficiency is the most important or critical issue in a data center administration it is usually selected as the main objective of interest. The other less important issues can be simply ignored or added as a constraint to the problem formulation. We found FFD the most common method to deal with the VM placement in its single objective from and it provides a sub-optimal solution with a worst-case limit. As we saw in this paper, many research works compare their proposed method to the FFD as a baseline.

When there are two or more equally important objectives that are needed to be minimized/maximized, we are encountering with intrinsic multi-objective VM placement problem. We observed that many existing schemes try to tackle multi-objective VM placement problem using weighted sum approach. Although the approach is simple and straightforward to use, it has some disadvantages as discussed earlier in this paper. In general, taking advantage of Pareto-based approaches is more recommended for tackling VM placement with multiple objectives since these approaches find the set of solutions that are optimal with subject to multiple objectives. As the search space for VM placement and particularly multi-objective VM placement problem as an NP-Hard problem is extremely large, the naïve (or exhaustive) search could be computationally expensive. Therefore, using meta-heuristics such as evolutionary techniques or swarm based intelligence methods are helpful and necessary for medium-sized to large data centers with a large number of PMs and VMs. These techniques try to find near-optimal solutions in a reasonable amount of time. CPU and memory are the two most common hardware resources considered in the literature. However, in some applications the other resources such as GPU and network equipment have a significant impact on the objective of VM placement and future work must have more concentration on these resources.

Acknowledgements This work was funded by the Institute of Research Management & Services (IPPP), University of Malaya.

References

1. Jing, S.-Y., Ali, S., She, K., Zhong, Y.: State-of-the-art research study for green cloud computing. J. Supercomput. **65**(1), 445–468 (2013). https://doi.org/10.1007/s11227-011-0722-1
2. Guo, Y., Fang, Y.: Electricity cost saving strategy in data centers by using energy storage. IEEE Trans. Parallel Distrib. Syst. **24**(6), 1149–1160 (2013)
3. Shigeta, S., Yamashima, H., Doi, T., Kawai, T., Fukui, K.: Design and implementation of a multi-objective optimization mechanism for virtual machine placement in cloud computing data center. In: Proceedings of the International Conference on Cloud Computing, pp. 21–31. Springer, Cham (2013)
4. Rasmussen, N.: Implementing energy efficient data centers. American Power Conversion, West Kingston (2006)
5. Guo, Y., Ding, Z., Fang, Y., Wu, D.: Cutting down electricity cost in internet data centers by using energy storage. In: Proceedings of the International Conference on IEEE Global Telecommunications Conference (GLOBECOM 2011), pp. 1–5. IEEE, Kathmandu (2011)
6. Dasgupta, G., Sharma, A., Verma, A., Neogi, A., Kothari, R.: Workload management for power efficiency in virtualized data centers. Commun. ACM **54**(7), 131–141 (2011)
7. Li, X., Qian, Z., Lu, S., Wu, J.: Energy efficient virtual machine placement algorithm with balanced and improved resource utilization in a data center. Math. Comput. Modell. **58**(5), 1222–1235 (2013)
8. Bilal, K., Malik, S.U.R., Khalid, O., Hameed, A., Alvarez, E., Wijaysekara, V., Irfan, R., Shrestha, S., Dwivedy, D., Ali, M., Khan, S.U.: A taxonomy and survey on green data center networks. Future Gener. Comput. Syst. **36**, 189–208 (2013). https://doi.org/10.1016/j.future.2013.07.006
9. Kansal, N.J., Chana, I.: Artificial bee colony based energy-aware resource utilization technique for cloud computing. Concurr. Comput. **27**(5), 1207–1225 (2014)
10. Yu, Y., Gao, Y.: Constraint programming-based virtual machines placement algorithm in datacenter. In: Proceedings of the International Conference on Intelligent Information Processing VI, pp. 295–304. Springer, Berlin (2012)
11. Bellur, U., Rao, C.S.: Optimal placement algorithms for virtual machines. http://arxiv.org/abs/1011.5064. (2010)
12. Xu, J., Fortes, J.: A multi-objective approach to virtual machine management in datacenters. Paper presented at the 8th ACM International Conference on Autonomic Computing, Karlsruhe, Germany (2011)
13. Usmani, Z., Singh, S.: A survey of virtual machine placement techniques in a cloud data center. Proc. Comput. Sci. **78**, 491–498 (2016)
14. Masdari, M., Nabavi, S.S., Ahmadi, V.: An overview of virtual machine placement schemes in cloud computing. J. Netw. Comput. Appl. **66**, 106–127 (2016)
15. Lopez-Pires, F., Baran, B.: Virtual machine placement literature review. http://arxiv.org/abs/1506.01509 (2015)
16. Pietri, I., Sakellariou, R.: Mapping virtual machines onto physical machines in cloud computing: a survey. ACM Comput. Surv. (CSUR) **49**(3), 49 (2016)
17. Liang, H., Xing, T., Cai, L.X., Huang, D., Peng, D., Liu, Y.: Adaptive computing resource allocation for mobile cloud computing. Int. J. Distrib. Sens. Netw. **2013**, 14 (2013). https://doi.org/10.1155/2013/181426
18. Subashini, S., Kavitha, V.: A survey on security issues in service delivery models of cloud computing. J. Netw. Comput. Appl. **34**(1), 1–11 (2011). https://doi.org/10.1016/j.jnca.2010.07.006
19. Do, T.V., Rotter, C.: Comparison of scheduling schemes for on-demand IaaS requests. J. Syst. Softw. **85**(6), 1400–1408 (2012)
20. Fei, X., Fangming, L., Hai, J., Vasilakos, A.V.: Managing performance overhead of virtual machines in cloud computing: a survey, state of the art, and future directions. Proc. IEEE **102**(1), 11–31 (2014). https://doi.org/10.1109/JPROC.2013.2287711
21. Kim, G., Park, H., Yu, J., Lee, W.: Virtual machines placement for network isolation in clouds. Paper presented at the ACM Research in Applied Computation Symposium, San Antonio, TX (2012)
22. Jeyarani, R., Nagaveni, N., Ram, R.V.: Self adaptive particle swarm optimization for efficient virtual machine provisioning in cloud. Int. J. Intell. Inf. Technol. (IJIIT) **7**(2), 25–44 (2011)
23. Graubner, P., Schmidt, M., Freisleben, B.: Energy-efficient virtual machine consolidation. IT Prof. **15**(2), 0028–0034 (2013)

24. Li, H., Wang, J., Peng, J., Wang, J., Liu, T.: Energy-aware scheduling scheme using workload-aware consolidation technique in cloud data centres. Commun. China **10**(12), 114–124 (2013). https://doi.org/10.1109/CC.2013.6723884
25. Vogels, W.: Beyond server consolidation. Queue **6**(1), 20–26 (2008)
26. Verma, A., Ahuja, P., Neogi, A.: Power-aware dynamic placement of hpc applications. Paper presented at the 22nd Annual International Conference on Supercomputing, Greece (2008)
27. Anand, A.: Adaptive Virtual Machine Placement supporting performance SLAs. Master's thesis, Supercomputer Education and Research Center, Indian Institute of Science (2013)
28. Medina, V., García, J.M.: A survey of migration mechanisms of virtual machines. ACM Comput. Surv. (CSUR) **46**(3), 30 (2014)
29. Wood, T., Shenoy, P., Venkataramani, A., Yousif, M.: Sandpiper: black-box and gray-box resource management for virtual machines. Comput. Netw. **53**(17), 2923–2938 (2009)
30. Gao, Y., Guan, H., Qi, Z., Wang, B., Liu, L.: Quality of service aware power management for virtualized data centers. J. Syst. Architect. **59**(4), 245–259 (2013)
31. Birkenheuer, G., Brinkmann, A., Kaiser, J., Keller, A., Keller, M., Kleineweber, C., Konersmann, C., Niehörster, O., Schäfer, T., Simon, J.: Virtualized HPC: a contradiction in terms. Software **42**(4), 485–500 (2012)
32. Pearce, M., Zeadally, S., Hunt, R.: Virtualization: issues, security threats, and solutions. ACM Comput. Surv. (CSUR) **45**(2), 17 (2013)
33. Kaplan, J.M., Forrest, W., Kindler, N.: Revolutionizing data center energy efficiency. In. Technical report, McKinsey & Company, New York (2008)
34. Luo, J.-P., Li, X., Chen, M.-R.: Hybrid shuffled frog leaping algorithm for energy-efficient dynamic consolidation of virtual machines in cloud data centers. Expert Syst. Appl. **41**(13), 5804–5816 (2014)
35. Buyya, R., Yeo, C.S., Venugopal, S., Broberg, J., Brandic, I.: Cloud computing and emerging IT platforms: vision, hype, and reality for delivering computing as the 5th utility. Future Gener. Comput. Syst. **25**(6), 599–616 (2009). https://doi.org/10.1016/j.future.2008.12.001
36. Gartner: Gartner Urges IT and Business Leaders to Wake up to IT's Energy Crisis. http://www.gartner.com/newsroom/id/496819 (2007). Accessed 2014
37. Gartner: Gartner estimates ICT industry accounts for 2 percent of global CO_2 emissions. http://www.gartner.com/newsroom/id/503867 (2007). Accessed 2014
38. Lee, Y.C., Zomaya, A.Y.: Energy efficient utilization of resources in cloud computing systems. J. Supercomput. **60**(2), 268–280 (2012)
39. Pascual, J.A., Lorido-Botrán, T., Miguel-Alonso, J., Lozano, J.A.: Towards a greener cloud infrastructure management using optimized placement policies. J. Grid Comput. (2014). https://doi.org/10.1007/s10723-014-9312-9
40. Lucas-Simarro, J.L., Moreno-Vozmediano, R., Montero, R.S., Llorente, I.M.: Scheduling strategies for optimal service deployment across multiple clouds. Future Gener. Comput. Syst. **29**(6), 1431–1441 (2013)
41. Ma, F., Liu, F., Liu, Z.: Multi-objective optimization for initial virtual machine placement in cloud data center. J. Inf. Comput. Sci. **9**(16), 5029–5038 (2012)
42. Zheng, Q., Li, R., Li, X., Shah, N., Zhang, J., Tian, F., Chao, K.-M., Li, J.: Virtual machine consolidated placement based on multi-objective biogeography-based optimization. Future Gener. Comput. Syst. **54**, 95–122 (2016). https://doi.org/10.1016/j.future.2015.02.010
43. Mastroianni, C., Meo, M., Papuzzo, G.: Probabilistic consolidation of virtual machines in self-organizing cloud data centers. IEEE Trans. Cloud Comput. **1**(2), 215–228 (2013). https://doi.org/10.1109/TCC.2013.17
44. Kanagavelu, R., Lee, B.-S., Le, N.T.D., Mingjie, L.N., Aung, K.M.M.: Virtual machine placement with two-path traffic routing for reduced congestion in data center networks. Comput. Commun. **53**, 1–12 (2014). https://doi.org/10.1016/j.comcom.2014.07.009
45. Speitkamp, B., Bichler, M.: A mathematical programming approach for server consolidation problems in virtualized data centers. IEEE Trans. Serv. Comput. **3**(4), 266–278 (2010). https://doi.org/10.1109/TSC.2010.25
46. Talbi, E.-G.: Metaheuristics: from design to implementation, vol. 74. Wiley, New Jersey (2009)
47. Tang, Z., Mo, Y., Li, K., Li, K.: Dynamic forecast scheduling algorithm for virtual machine placement in cloud computing environment. J. Supercomput. **70**(3), 1279–1296 (2014). https://doi.org/10.1007/s11227-014-1227-5
48. Liu, X.F., Zhan, Z.H., Deng, J.D., Li, Y., Gu, T., Zhang, J.: An energy efficient ant colony system for virtual machine placement in cloud computing. IEEE Trans. Evol. Comput. (2016). https://doi.org/10.1109/tevc.2016.2623803
49. Ajiro, Y., Tanaka, A.: Improving packing algorithms for server consolidation. In: Proceedings of the International Conference for the Computer Measurement Group (CMG), pp. 399–406 (2007)
50. Wilcox, D., McNabb, A., Seppi, K.: Solving virtual machine packing with a reordering grouping genetic algorithm. Paper Presented at the IEEE Congress of Evolutionary Computation (CEC), (2011)
51. Yan, J., Zhang, H., Xu, H., Zhang, Z.: Discrete PSO-based workload optimization in virtual machine placement. Pers. Ubiquit. Comput. **22**(3), 589–596 (2018)
52. Beloglazov, A., Abawajy, J., Buyya, R.: Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. Future Gener. Comput. Syst. **28**(5), 755–768 (2012)
53. Fan, X., Weber, W.-D., Barroso, L.A.: Power provisioning for a warehouse-sized computer. Paper Presented at the 34th annual international symposium on Computer architecture, San Diego, California, USA (2007)
54. Beloglazov, A., Buyya, R.: Adaptive threshold-based approach for energy-efficient consolidation of virtual machines in cloud data centers. Paper presented at the 8th International Workshop on Middleware for Grids, Clouds and e-Science, Bangalore, India (2010)
55. Beloglazov, A., Buyya, R.: Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers. Concurr. Comput. Pract. Exp. **24**(13), 1397–1420 (2012). https://doi.org/10.1002/cpe.1867
56. Quang-Hung, N., Nien, P.D., Nam, N.H., Tuong, N.H., Thoai, N.: A genetic algorithm for power-aware virtual machine allocation in private cloud. In: Proceedings of the International Conference on Information and Communication Technology, pp. 183–191. Springer, Berlin (2013)
57. Wang, X., Liu, X., Fan, L., Jia, X.: A decentralized virtual machine migration approach of data centers for cloud computing. Math. Probl. Eng. **2013**, 10 (2013). https://doi.org/10.1155/2013/878542
58. Ding, Y., Qin, X., Liu, L., Wang, T.: Energy efficient scheduling of virtual machines in cloud with deadline constraint. Future Gener. Comput. Syst. **50**, 62–74 (2015). https://doi.org/10.1016/j.future.2015.02.001
59. Lovász, G., Niedermeier, F., de Meer, H.: Performance tradeoffs of energy-aware virtual machine consolidation. Clust. Comput.

16(3), 481–496 (2013). https://doi.org/10.1007/s10586-012-0214-y
60. Madhusudhan, B., Sekaran, K.C.: A Genetic algorithm approach for virtual machine placement in cloud. Paper presented at the international conference on emerging research in computing, information, communication and applications (ERCICA 2013), Bangalore, India (2013)
61. Ebrahimirad, V., Goudarzi, M., Rajabi, A.: Energy-aware scheduling for precedence-constrained parallel virtual machines in virtualized data centers. J. Grid Comput. **13**(2), 233–253 (2015). https://doi.org/10.1007/s10723-015-9327-x
62. Verma, A., Ahuja, P., Neogi, A.: pMapper: power and migration cost aware application placement in virtualized systems. Paper presented at the 9th ACM/IFIP/USENIX international conference on the middleware, Leuven, Belgium (2008)
63. Abdullah, M., Lu, K., Wieder, P., Yahyapour, R.: A heuristic-based approach for dynamic VMS consolidation in cloud data centers. Arab. J. Sci. Eng. **1**, 15 (2017)
64. Gao, Y., Guan, H., Qi, Z., Song, T., Huan, F., Liu, L.: Service level agreement based energy-efficient resource management in cloud data centers. Comput. Electr. Eng. **40**(5), 1621–1633 (2014). https://doi.org/10.1016/j.compeleceng.2013.11.001
65. Kessaci, Y., Melab, N., Talbi, E.-G.: A multi-start local search heuristic for an energy efficient VMs assignment on top of the OpenNebula cloud manager. Future Gener. Comput. Syst. **36**, 237–256 (2014)
66. Milojičić, D., Llorente, I.M., Montero, R.S.: Opennebula: a cloud management tool. IEEE Internet Comput. **15**(2), 11–14 (2011)
67. Ferreto, T.C., Netto, M.A., Calheiros, R.N., De Rose, C.A.: Server consolidation with migration control for virtualized data centers. Future Gener. Comput. Syst. **27**(8), 1027–1034 (2011)
68. Alharbi, F., Tian, Y.-C., Tang, M., Zhang, W.-Z., Peng, C., Fei, M.: An ant colony system for energy-efficient dynamic virtual machine placement in data centers. Expert Syst. Appl. **120**, 228–238 (2019)
69. Liu, X.-F., Zhan, Z.-H., Du, K.-J., Chen, W.-N.: Energy aware virtual machine placement scheduling in cloud computing based on ant colony optimization approach. Paper presented at the Genetic and evolutionary computation, Vancouver, BC, Canada (2014)
70. Alharbi, F., Tian, Y.-C., Tang, M., Ferdaus, M.H.: Profile-based ant colony optimization for energy-efficient virtual machine placement. In: Proceedings of the International Conference on Neural Information Processing 2017, pp. 863–871. Springer, Cham (2017)
71. Xiao, Z., Ming, Z.: A state based energy optimization framework for dynamic virtual machine placement. Data Knowl. Eng. **120**, 83–99 (2019)
72. Greenberg, A., Hamilton, J., Maltz, D.A., Patel, P.: The cost of a cloud: research problems in data center networks. ACM SIGCOMM Comput. Commun. Rev. **39**(1), 68–73 (2008)
73. Fang, W., Liang, X., Li, S., Chiaraviglio, L., Xiong, N.: VMPlanner: optimizing virtual machine placement and traffic flow routing to reduce network power costs in cloud data centers. Comput. Netw. **57**(1), 179–196 (2013)
74. Liu, X., Gu, H., Zhang, H., Liu, F., Chen, Y., Yu, X.: Energy-Aware on-chip virtual machine placement for cloud-supported cyber-physical systems. Microprocess. Microsyst. **52**, 427–437 (2017). https://doi.org/10.1016/j.micpro.2016.07.013
75. Meng, X., Pappas, V., Zhang, L.: Improving the scalability of data center networks with traffic-aware virtual machine placement. Paper presented at the 29th conference on Information communications, San Diego, California, USA (2010)
76. Armour, G.C., Buffa, E.S.: A heuristic algorithm and simulation approach to relative location of facilities. Manage. Sci. **9**(2), 294–309 (1963)
77. Burkard, R.E., Rendl, F.: A thermodynamically motivated simulation procedure for combinatorial optimization problems. Eur. J. Oper. Res. **17**(2), 169–174 (1984)
78. da Silva, R.A.C., da Fonseca, N.L.S.: Topology-aware virtual machine placement in data centers. J. Grid Comput. **14**(1), 75–90 (2016). https://doi.org/10.1007/s10723-015-9343-x
79. Rahimzadeh Ilkhechi, A., Korpeoglu, I., Ulusoy, Ö.: Network-aware virtual machine placement in cloud data centers with multiple traffic-intensive components. Comput. Netw. **91**, 508–527 (2015). https://doi.org/10.1016/j.comnet.2015.08.042
80. Song, F., Huang, D., Zhou, H., Zhang, H., You, I.: An optimization-based scheme for efficient virtual machine placement. Int. J. Parallel Prog. **42**(5), 853–872 (2013)
81. Xu, J., Fortes, J.A.: Multi-objective virtual machine placement in virtualized data center environments. Paper presented at the IEEE/ACM international conference on green computing and communications (GreenCom) and IEEE/ACM international conference on cyber, physical and social computing (CPSCom), Hangzhou, China (2010)
82. Cho, K.-M., Tsai, P.-W., Tsai, C.-W., Yang, C.-S.: A hybrid meta-heuristic algorithm for VM scheduling with load balancing in cloud computing. Neural Comput. Appl. **26**(6), 1297–1309 (2014). https://doi.org/10.1007/s00521-014-1804-9
83. He, L., Zou, D., Zhang, Z., Chen, C., Jin, H., Jarvis, S.A.: Developing resource consolidation frameworks for moldable virtual machines in clouds. Future Gener. Comput. Syst. **32**, 69–81 (2014). https://doi.org/10.1016/j.future.2012.05.015
84. Hermenier, F., Lorca, X., Menaud, J.-M., Muller, G., Lawall, J.: Entropy: a consolidation manager for clusters. Paper presented at the ACM SIGPLAN/SIGOPS international conference on virtual execution environments, Washington, DC, USA (2009)
85. Wray, M.: From server consolidation to network consolidation. Netw. Secur. **2012**(2), 8–11 (2012). https://doi.org/10.1016/S1353-4858(12)70014-4
86. Khosravi, A., Garg, S., Buyya, R.: Energy and carbon-efficient placement of virtual machines in distributed cloud data centers. In: Wolf, F., Mohr, B., Mey, D. (eds.) Euro-Par 2013 Parallel Processing. Lecture Notes in Computer Science, vol. 8097, pp. 317–328. Springer, Berlin (2013)
87. Moghaddam, F.F., Moghaddam, R.F., Cheriet, M.: Carbon-aware distributed cloud: multi-level grouping genetic algorithm. Clust.Comput. (2014). https://doi.org/10.1007/s10586-014-0359-y
88. Pop, C.B., Anghel, I., Cioara, T., Salomie, I., Vartic, I.: A swarm-inspired data center consolidation methodology. Paper presented at the 2nd international conference on web intelligence, mining and semantics, Craiova, Romania (2012)
89. Son, S., Jung, G., Jun, S.: An SLA-based cloud computing that facilitates resource allocation in the distributed data centers of a cloud provider. J. Supercomput. **64**(2), 606–637 (2013). https://doi.org/10.1007/s11227-012-0861-z
90. Tordsson, J., Montero, R.S., Moreno-Vozmediano, R., Llorente, I.M.: Cloud brokering mechanisms for optimized placement of virtual machines across multiple providers. Future Gener. Comput. Syst. **28**(2), 358–367 (2012)
91. Fourer, R., Gay, D.M., Kernighan, B.W.: A modeling language for mathematical programming. Manage. Sci. **36**(5), 519–554 (1990)
92. IBM Corporation: CPLEX Optimizer. http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/index.html. Accessed Oct 2014

93. Dongarra, J.J., Luszczek, P., Petitet, A.: The LINPACK benchmark: past, present and future. Concurr. Comput. Pract. Exp. **15**(9), 803–820 (2003)
94. Fourer, R., Gay, D.M., Kernighan, B.W.: AMPL: A Mathematical Programming Language. AT&T Bell Laboratories, Murray Hill (1987)
95. Gao, Y., Guan, H., Qi, Z., Hou, Y., Liu, L.: A multi-objective ant colony system algorithm for virtual machine placement in cloud computing. J. Comput. Syst. Sci. **79**(8), 1230–1242 (2013)
96. Deb, K.: Multi-objective optimization. In: Burke, E.K., Kendall, G. (eds.) Search Methodologies, pp. 403–449. Springer, New York (2014)
97. Gen, M., Cheng, R.: Genetic Algorithm and Engineering Optimization. Wiley, New York (2000)
98. Caponio, A., Neri, F.: Integrating cross-dominance adaptation in multi-objective memetic algorithms. In: Goh, C.-K., Ong, Y.-S., Tan, K. (eds.) Multi-Objective Memetic Algorithms. Studies in Computational Intelligence, vol. 171, pp. 325–351. Springer, Berlin (2009)
99. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. Evol. Comput. **6**(2), 182–197 (2002)
100. Feller, E., Rilling, L., Morin, C.: Energy-aware ant colony based workload placement in clouds. Paper presented at the 12th IEEE/ACM international conference on grid computing, Lyon (2011)
101. Veldhuizen, D.: Multiobjective evolutionary algorithms: classifications, analyses, and new innovations. In: School of Engineering of the Air Force Institute of Technology, Dayton, Ohio (1999)
102. Schott, J.R.: Fault tolerant design using single and multicriteria genetic algorithm optimization. In: Air Force Inst of Tech Wright-Patterson AFB OH (1995)
103. Jamali, S., Malektaji, S., Analoui, M.: An imperialist competitive algorithm for virtual machine placement in cloud computing. J. Exp. Theor. Artif. Intell. **29**(3), 575–596 (2017)
104. Atashpaz-Gargari, E., Lucas, C.: Imperialist competitive algorithm: an algorithm for optimization inspired by imperialistic competition. Paper presented at the IEEE congress on evolutionary eomputation. CEC (2007)
105. Sharifi, M., Salimi, H., Najafzadeh, M.: Power-efficient distributed scheduling of virtual machines using workload-aware consolidation techniques. J. Supercomput. **61**(1), 46–66 (2012)
106. Dong, J., Wang, H., Li, Y., Cheng, S.: Virtual machine scheduling for improving energy efficiency in IaaS cloud. Commun. China **11**(3), 1–12 (2014). https://doi.org/10.1109/CC.2014.6825253
107. Tang, M., Pan, S.: A hybrid genetic algorithm for the energy-efficient virtual machine placement problem in data centers. Neural Process. Lett. **41**(2), 211–221 (2014)
108. Chen, X., Jiang, J.-H.: A method of virtual machine placement for fault-tolerant cloud applications. Intell. Autom. Soft Comput. **22**(4), 587–597 (2016). https://doi.org/10.1080/10798587.2016.1152775
109. Gupta, M.K., Amgoth, T.: Resource-aware virtual machine placement algorithm for IaaS cloud. J. Supercomput. **74**(1), 122–140 (2018)
110. Esfandiarpoor, S., Pahlavan, A., Goudarzi, M.: Structure-aware online virtual machine consolidation for datacenter energy improvement in cloud computing. Comput. Electr. Eng. **42**, 74–89 (2015)
111. Calheiros, R.N., Ranjan, R., Beloglazov, A., De Rose, C.A., Buyya, R.: CloudSim a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Softw. Pract. Exp. **41**(1), 23–50 (2011)
112. Yue, M.: A simple proof of the inequality FFD (L) $\leq$ 11/9 OPT (L) + 1,$\forall$ L for the FFD bin-packing algorithm. Acta Mathematicae Applicatae Sinica **7**(4), 321–331 (1991)
113. Zhao, H., Wang, J., Liu, F., Wang, Q., Zhang, W., Zheng, Q.: Power-aware and performance-guaranteed virtual machine placement in the cloud. IEEE Trans. Parallel Distrib. Syst. **29**(6), 1385–1400 (2018)
114. Wang, J., Huang, C., He, K., Wang, X., Chen, X., Qin, K.: An energy-aware resource allocation heuristics for VM scheduling in cloud. In: Proceedings of the 2013 International Conference on IEEE 10th International Conference on High Performance Computing and Communications & 2013 IEEE International Conference on Embedded and Ubiquitous Computing, pp. 587–594. IEEE (2013)
115. Bobroff, N., Kochut, A., Beaty, K.: Dynamic placement of virtual machines for managing sla violations. Paper presented at the 10th IFIP/IEEE international symposium on integrated network management, Munich (2007)
116. Khargharia, B., Hariri, S., Yousif, M.S.: Autonomic power and performance management for computing systems. Clust. Comput. **11**(2), 167–181 (2008)
117. Ranganathan, P., Leech, P., Irwin, D., Chase, J.: Ensemble-level power management for dense blade servers. Paper presented at the ACM SIGARCH computer architecture news (2006)
118. Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: Improving the strength Pareto evolutionary algorithm. In: Proceedings of the International Conference on Swiss Federal Institute of Technology (ETH), Zurich, Switzerland (2001)
119. Bianchini, R., Rajamony, R.: Power and energy management for server systems. IEEE Comput. **37**(11), 68–74 (2004)
120. Wu, G., Tang, M., Tian, Y.-C., Li, W.: Energy-efficient virtual machine placement in data centers by genetic algorithm. In: Huang, T., Zeng, Z., Li, C., Leung, C. (eds.) Neural Information Processing. Lecture Notes in Computer Science, vol. 7665, pp. 315–323. Springer, Berlin (2012)
121. Chen, T., Gao, X., Chen, G.: Optimized virtual machine placement with traffic-aware balancing in data center networks. Sci. Programm. **2016**, 10 (2016). https://doi.org/10.1155/2016/3101658
122. Gupta, A., Milojicic, D., Kalé, L.V.: Optimizing VM placement for HPC in the cloud. Paper presented at the workshop on cloud services, federation, and the 8th open cirrus summit, San Jose, California, USA (2012)
123. Gupta, A., Kalé, L.V., Milojicic, D., Faraboschi, P., Balle, S.M.: HPC-Aware VM Placement in Infrastructure Clouds. Paper presented at the IEEE international conference on cloud engineering (IC2E), Redwood City, CA (2013)
124. OpenStack Open Source Cloud Computing Software. https://www.openstack.org
125. Avetisyan, A.I., Campbell, R., Gupta, I., Heath, M.T., Ko, S.Y., Ganger, G.R., Kozuch, M.A., O'Hallaron, D., Kunze, M., Kwan, T.T., Lai, K., Lyons, M., Milojicic, D.S., Hing Yan, L., Yeng Chai, S., Ng Kwang, M., Luke, J.Y., Han, N.: Open cirrus: a global cloud computing testbed. Computer **43**(4), 35–43 (2010). https://doi.org/10.1109/MC.2010.111
126. Jin, H., Qin, H., Wu, S., Guo, X.: CCAP: a cache contention-aware virtual machine placement approach for hpc cloud. Int. J. Parallel Prog. **43**(3), 403–420 (2013). https://doi.org/10.1007/s10766-013-0286-1
127. Kim, S.-G., Eom, H., Yeom, H.: Virtual machine consolidation based on interference modeling. J. Supercomput. **66**(3), 1489–1506 (2013). https://doi.org/10.1007/s11227-013-0939-2
128. Mc Evoy, G., Mury, A.R., Schulze, B.: An analysis of definition and placement of virtual machines for high performance applications on Clouds. Concurr. Comput. Pract. Exp. **27**(7), 1789–1814 (2014). https://doi.org/10.1002/cpe.3346

129. Stillwell, M., Vivien, F., Casanova, H.: Virtual machine resource allocation for service hosting on heterogeneous distributed platforms. Paper presented at the 26th IEEE international parallel and distributed processing symposium, Shanghai, China (2012)
130. Lucas Simarro, J.L., Moreno-Vozmediano, R., Montero, R.S., Llorente, I.M.: Dynamic placement of virtual machines for cost optimization in multi-cloud environments. Paper presented at the international conference on high performance computing and simulation (HPCS), Istanbul (2011)
131. Chaisiri, S., Lee, B.-S., Niyato, D.: Optimal virtual machine placement across multiple cloud providers. Paper presented at the IEEE Asia-Pacific services computing conference, Singapore (2009)
132. Lucas-Simarro, J.L., Moreno-Vozmediano, R., Montero, R.S., Llorente, I.M.: Cost optimization of virtual infrastructures in dynamic multi-cloud scenarios. Concurr. Comput. Pract. Exp. **27**(9), 2260–2277 (2012). https://doi.org/10.1002/cpe.2972
133. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms. MIT Press, Cambridge (2001)
134. Michalewicz, Z., Fogel, D.B.: How to Solve It: Modern Heuristics. Springer Science & Business Media, New York (2004)
135. Perumal, V., Subbiah, S.: Power-conservative server consolidation based resource management in cloud. Int. J. Netw. Manage **24**(6), 415–432 (2014). https://doi.org/10.1002/nem.1873
136. Hillier, M., Hillier, F.: Conventional optimization techniques. In: Sarker, R., et al. (eds.) Evolutionary Optimization. International Series in Operations Research & Management Science, pp. 3–25. Springer, New York (2002)
137. Hillier, F.S., Lieberman, G.J.: Introduction to operations research. Tata McGraw-Hill Education, New York (2001)
138. Holland, J.H.: Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. University of Michigan Press, Ann Arbor (1975)
139. Dorigo, M., Birattari, M.: Ant colony optimization. In: Sammut, C., Webb, G.I. (eds.) Encyclopedia of Machine Learning, pp. 36–39. Springer, New York (2010)
140. Blum, C., Roli, A.: Hybrid metaheuristics: an introduction. In: Blum, C., Roli, A. (eds.) Hybrid Metaheuristics, pp. 1–30. Springer, New York (2008)
141. Dorigo, M., Blum, C.: Ant colony optimization theory: a survey. Theoret. Comput. Sci. **344**(2), 243–278 (2005)
142. Dowsland, K.A., Thompson, J.M.: Simulated annealing. In: Popovici, E., et al. (eds.) Handbook of Natural Computing. Springer, New York (2012)
143. Russell, S.J., Norvig, P.: Artificial Intelligence: A Modern Approach. Pearson Education, London (2003)
144. Henderson, D., Jacobson, S., Johnson, A.: The Theory and Practice of Simulated Annealing. In: Glover, F., Kochenberger, G. (eds.) Handbook of Metaheuristics. International Series in Operations Research & Management Science, vol. 57, pp. 287–319. Springer, New York (2003)
145. Karaboga, D.: An idea based on honey bee swarm for numerical optimization. In. Technical Report-tr06, Erciyes University, Engineering Faculty, Computer Engineering Department (2005)
146. Karaboga, D., Basturk, B.: Artificial bee colony (ABC) optimization algorithm for solving constrained optimization problems. In: Proceedings of the International Conference on 12th International Fuzzy Systems Association World Congress. Springer, New York (2007)
147. Glover, F.: Future paths for integer programming and links to artificial intelligence. Comput. Oper. Res. **13**(5), 533–549 (1986)
148. Moscato, P.: On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. Caltech concurrent computation program, C3P Report 826, 1989 (1989)
149. Donoso, Y., Fabregat, R.: Multi-objective optimization in computer networks using metaheuristics. Auerbach Publications, Boca Raton (2016)
150. Yu, X., Gen, M.: Introduction to evolutionary algorithms. Springer, New York (2010)
151. Merz, P., Freisleben, B.: A comparison of memetic algorithms, tabu search, and ant colonies for the quadratic assignment problem. In: Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406) 1999, pp. 2063–2070. IEEE
152. Elbeltagi, E., Hegazy, T., Grierson, D.: Comparison among five evolutionary-based optimization algorithms. Adv. Eng. Inform. **19**(1), 43–53 (2005)
153. Yue, W., Chen, Q.: Dynamic placement of virtual machines with both deterministic and stochastic demands for green cloud computing. Math. Probl. Eng. (2014). https://doi.org/10.1155/2014/613719
154. Ming, C., Hui, Z., Ya-Yunn, S., Xiaorui, W., Guofei, J., Yoshihira, K.: Effective VM sizing in virtualized data centers. Paper presented at the IFIP/IEEE international symposium on integrated network management, Dublin (2011)
155. Benson, T., Akella, A., Maltz, D.A.: Network traffic characteristics of data centers in the wild. Paper presented at the 10th ACM SIGCOMM conference on Internet measurement, Melbourne, Australia (2010)
156. Kandula, S., Sengupta, S., Greenberg, A., Patel, P., Chaiken, R.: The nature of data center traffic: measurements & analysis. Paper presented at the 9th ACM SIGCOMM internet measurement conference, Chicago, Illinois, USA (2009)
157. Jin, H., Pan, D., Xu, J., Pissinou, N.: Efficient VM placement with multiple deterministic and stochastic resources in data centers. Paper presented at the IEEE Global Communications Conference (GLOBECOM), Anaheim, CA (2012)
158. Meng, W., Xiaoqiao, M., Li, Z.: Consolidating virtual machines with dynamic bandwidth demand in data centers. Paper presented at the IEEE INFOCOM, Shanghai (2011)
159. Isci, C., Hanson, J.E., Whalley, I., Steinder, M., Kephart, J.O.: Runtime Demand Estimation for effective dynamic resource management. Paper presented at the IEEE Network Operations and Management Symposium (NOMS), Osaka (2010)
160. Beloglazov, A.: Energy-efficient management of virtual machines in data centers for cloud computing. The University of Melbourne, Parkville (2013)

Hamid Talebian obtained his bachelor and master of computer engineering in 2003 and 2007 respectively from Azad University. In 2013, he completed his PhD of Computer Science from the University of Malaya. In 2013, he joined Center for Mobile Cloud Computing Research (C4MCCR) in University of Malaya. His research interest includes cloud computing security, evolutionary computation, and multi-objective optimization.

Abdullah Gani is Professor at the Dept. of Computer System and Technology, Faculty of Computer Science and Information Technology, University of Malaya, Malaysia. His academic qualifications were obtained from the University of Hull, UK, for bachelor and master degrees, and the University of Sheffield, UK, for Ph.D. He has vast teaching experience due to having worked in various educational institutions locally and abroad—schools, teaching college, ministry of education, and universities. His interest in research started in 1983 when he was chosen to attend the Scientific Research Course in RECSAM by the Ministry of Education, Malaysia. More than 100 academic papers have been published in conferences and respectable journals. He actively supervises many students at all levels of study—Bachelor, Master, and Ph.D. His interest of research includes self-organized system, reinforcement learning, and wireless-related networks. He is now working on mobile cloud computing with High Impact Research Grant of USD 500,000 (RM 1.5M) for the period of 2011–2016. He is a senior member of IEEE. Currently, he is a director of the Centre for Mobile Cloud Computing Research, which focuses on high impact research. He is also a visiting Professor at the King Saud University, Saudi Arabia as well as serves as Adjunct Professor at the COMSATS Institute of Information Technology, Islamabad, Pakistan. He also serves as a visiting professor at the University Malaysia Sabah, Kota Kinabalu, Sabah, Malaysia (2015–2017). He serves as a chairman of Industry Advisory Panel for Research Degree Program at UNITEN, Malaysia (2015–2017).

Mehdi Sookhak received the B.Sc. degree in computer engineering from Shiraz University, Iran, in 2001, and the M.Sc. degree in Computer Science (information security) from the University Technology Malaysia. He completed his Ph.D. in 2015 from University of Malaya. His research interest includes Data Storage Security, Cloud Computing Security, and Digital Forensics.

Ahmed Abdelaziz Abdelatif obtained his M.S. (2007) and Ph.D. (2017) degrees in Computer Science and Information Technology at the University of Malaya(UM), Malaysia. He has been involved in Centre for Mobile Cloud Computing Research (C4MCCR) projects funded by Malaysian Ministry of Higher Education. In his Ph.D. research, Ahmed proposed a novel service based load balancing in the cloud using SDN and OpenStack. He published numbers of the ISI index papers in the area of the SDN, OpenFlow and Network Virtualization. Currently, he is a full-time Assistant Professor in the Future University (FU) in Sudan. His areas of interest include SDN/NFV Technology, OpenStack, Network virtualization. Ahmed works on ONOS and OpenStack since October 2015 during his Ph.D. Research Project.

Abdullah Yousafzai obtained his Ph.D. from the University of Malaya in 2017. He received his BCS (Hons) and MS (Computer Science) from Hazara University Mansehra, Pakistan, and Comsats Institute of Information Technology, Abbottabad, in 2009 and 2013, respectively. In past, he worked as a Web Developer and is currently associated as a Research Assistant in the Center for Mobile Cloud Computing Research (C4MCCR). His research work focus is mainly on Resource Management in Parallel and Distributed Environments.

Athanasios V. Vasilakos is currently Professor at University of Western Macedonia, Greece. He has authored or co-authored over 200 technical papers in major International Journals and Conferences. He is author/coauthor of five books and 20 book chapters in the areas of communications. Prof. Vasilakos has served as General Chair, TPC Chair for many conferences. He has served as an Editor or/and Guest Editor for many technical journals, such as the IEEE TIFS, IEEE TCC, IEEE Transactions on Network and Services Management, IEEE Transactions on Systems, Man, and Cybernetics, IEEE Transactions on Information Technology in Biomedicine, IEEE Transactions on Computers, ACM trans. on autonomous and adaptive systems, IEEE JSAC special issues of May 2009, Jan. 2011, March 2011, IEEE Commag, ACM WINET, ACM MONET. He is founding Editor-in-Chief of the IJAACS and the IJART. He is General Chair of the Council of Computing of the European Alliances for Innovation.

Fei Richard Yu received the Ph.D. degree in electrical engineering from the University of British Columbia (UBC) in 2003. From 2002 to 2006, he was with Ericsson (in Lund, Sweden) and a start-up (in San Diego, CA, USA), where he worked on the research and development in the areas of advanced wireless communication technologies and new standards. He joined Carleton School of Information Technology and the Department of Systems and Computer Engineering (cross-appointment) at Carleton University, Ottawa, in 2007, where he is currently a Professor. His research interests include connected and autonomous vehicles, wireless cyber-physical systems, security and privacy, and machine learning and artificial intelligence. He has published 480 + papers in reputable journals/conferences, 6 edited books, and 27 granted patents, with 10,000 + citations (Google Scholar).