An ACO for Energy-Efficient and Traffic-Aware Virtual Machine Placement in Cloud Computing

Authors:
Huanlai Xing [1], Jing Zhu [1], Rong Qu [2], Penglin Dai [1], Shouxi Luo [1], Muhammad Azhar Iqbal [1]
Affiliation:

[1] School of Information Science and Technology, Southwest Jiaotong University, P. R. China
[2] COL Lab, School of Computer Science, University of Nottingham, United Kingdom
Corresponding author: Huanlai Xing
Telephone/E-mail: +(86) 18780272961/ hxx@home.swjtu.edu.cn
Address: Room 9439, Teaching building 9, School of Information Science and Technology, Southwest Jiaotong University (Xipu Campus), Chengdu 611756, China

***Abstract*:**

This paper formulates a virtual machine placement (VMP) problem, where the total power consumption of physical machines (PMs) and switches and the total network bandwidth resource consumption among VMs are jointly minimized. To address the problem, we present an energy- and traffic-aware ant colony optimization (ETA-ACO) algorithm. Three novel schemes are introduced to enhance the performance of ETA-ACO, including an energy- and bandwidth-aware PM selection scheme, a traffic-based VM ordering scheme, and a direct information exchange scheme. The first scheme consists of two steps when selecting a PM to host a given VM. In the first step, PMs with lower power consumption are preserved. In the second step, the one with the lowest bandwidth resource consumption is chosen to host the VM. In the second scheme, ETA-ACO places VMs in descending order by their traffic demands. The third scheme constructs new solutions by spreading the components of the best solution over a group of constructed solutions. Simulation results demonstrate that the three novel schemes are effective in adapting ETA-ACO for the VMP problem. Besides, ETA-ACO outperforms a number of state-of-the-art heuristics and metaheuristics in terms of solution quality.

***Keywords*:** Ant colony optimization; cloud computing; evolutionary computation; virtual machine placement

1. Introduction

Nowadays, datacenters have become larger and larger, leading to a dramatic increase in power consumption and cooling expenses. There are mainly three types of equipment in a datacenter that consume electricity, namely physical machines (i.e., servers), cooling systems, and network devices. They usually consume 40-55%, 15-30%, and 10-25% of the total power, respectively [1].

As one of the supporting technologies in datacenters, virtualization abstracts physical resources as logical resources in a flexible manner [2]. In a virtualized datacenter, applications may be run on virtual machines (VMs) or containers. Enabling efficient pooling of hardware resources, such as CPU, RAM, and I/O interfaces, allows a physical machine (PM) to accommodate multiple VMs. With on-demand and dynamic provision, the virtualization technique improves the utilization of physical resources and reduces energy consumption.

An appropriate set of PMs for hosting VMs consumes less energy and improves datacenter efficiency. Hence, it is vital to optimizing the placement of VMs, referred to as the virtual machine placement (VMP) problem. VMP is one of the critical issues in resource management and allocation in datacenters. Unfortunately, this problem is NP-hard [3], requiring non-trivial computational time for deterministic mathematical methods to find the optimal VM placement. So far, a number of challenging issues have been emphasized when studying VMP problems, e.g., energy saving on PMs, traffic flow routing, network resource consumption, and so on. Nevertheless, to the best of our knowledge, research on jointly optimizing power consumption and network resource utilization has not received enough attention (see Subsection 2.1 for details).

Metaheuristics are a family of nature-inspired stochastic search algorithms capable of achieving high-quality solutions within limited computational time, especially suitable for solving NP-hard problems [4][5]. These algorithms have demonstrated their excellent potential for addressing complex real-world engineering problems, such as image classification [6], feature selection [7], task scheduling [8], sensor node deployment [9], and vehicle routing [10]. In recent research on metaheuristics, simulated annealing (SA), genetic algorithm (GA), and particle swarm optimization (PSO), etc. have been used in optimizing VMP problems in terms of energy consumption, quality-of-service (QoS), resource utilization, and so on [11][12]. Ant colony optimization (ACO) is a population-based probabilistic search algorithm mimicking the foraging behavior of ants [13]. Each ant seeks a short path between its nest and a food source in ACO by exploiting the pheromone deposited before. Because of its effective distributed positive feedback mechanism and strong parallel computing ability, ACO has been successfully applied to various optimization problems such as traveling salesman problem [14], network coding resource minimization [15], railway junction rescheduling [16],

network routing problem [17], as well as the VMP problem [18][19].

This paper addresses the VMP problem considering both power and network resource consumption. An energy- and traffic-aware ACO (ETA-ACO) with three performance-enhancing schemes is proposed and investigated. The results demonstrate that ETA-ACO outperforms a number of state-of-the-art heuristics and metaheuristic algorithms in terms of solution quality. The contributions of our work are listed as follows.

- A new variant of the VMP problem is formulated, where energy saving on PMs and switches and bandwidth resource consumption of links are both considered. The datacenter is based on the *Fat-Tree* topology, with heterogeneous switches and PMs deployed. For an arbitrary switch, each port can support a number of data rate modes. Different ports working at different data rate modes lead to different amounts of power consumption. In the VMP problem, the power consumption by PMs and switches and the bandwidth resource consumption by communication among VMs are minimized simultaneously.
- ACO is adapted for the new VMP problem, with three performance-enhancing schemes. These schemes include an energy- and bandwidth-aware PM selection scheme, a traffic-based VM ordering scheme, and a direct information exchange scheme. The first scheme takes energy-saving and bandwidth consumption into account in the process of PM selection. The second one provides the search with a set of VMs ordered according to the traffic demands, reducing the bandwidth resource consumption. The third enables faster information exchange among already constructed solutions, which complements the pheromone-based indirect information exchange among ants in ACO.

The rest of the paper is organized as follows: Section 2 introduces the problem formulation and overviews the related work. Section 3 reviews a traditional ACO for the VMP problem. In Section 4, the proposed ETA-ACO is described in detail. Section 5 justifies the performance of the proposed algorithm based on evaluations and comparisons against state-of-the-art approaches. Section 6 concludes the paper.

2. Related Work and Problem Definition

2.1 Related Work

With the rapid development in cloud computing, an increasing amount of research efforts have been dedicated to VMP problems.

Power consumption is a mainstream concern. Fang *et al.* presented a prototype VM placement approach, VMPlanner, to reduce the consumed network power in datacenters [20]. They were concerned with two tasks, namely

VM placement and traffic flow routing, by switching off as many unneeded network devices as possible for energy-saving purposes. Gaggero *et al*. designed a holistic placement framework, with conflicting performance indicators considered, e.g., the service level, the energetic footprint, and security policies [21]. The authors managed the VM-PM mapping via model predictive control. Farahnakian *et al*. studied the least energy consumption VM consolidation problem [22]. A regression-based model was devised to estimate the future CPU and memory utilization of VMs and PMs. Chen *et al*. presented a two-tier VMP framework for reducing power consumption and accommodating as many VMs as possible [23]. Hieu *et al*. improved the datacenter's energy efficiency via resource utilization prediction, where the long-term usage of physical resources was estimated in advance [24].

Network traffic reduction saves bandwidth resources. Ilkhechi *et al.* formulated a metric called satisfaction, indicating a VM's performance on a particular PM, and introduced a greedy algorithm and a heuristic-based algorithm considering the communication pattern and flow demand of VMs [25]. Fukunaga *et al.* minimized the VMP connection cost in a datacenter, where the total span connecting all PMs hosting VMs was taken into account [26]. They developed centralized and distributed models for the problem and showed that the proposed algorithms could provide a high-quality approximation to the optima. Wang *et al*. developed a network-aware VM ensemble placement system, MAPLE [27]. By empirically estimating the bandwidth resource required by servers, MAPLE provided satisfied quality-of-service (QoS) services for multi-tenant applications. Guo *et al*. investigated the min-max datacenter load problem based on shadow routing, where an algorithm combining VM routing and VM-to-PM placement was devised [28]. Son *et al*. proposed a priority-aware VMP algorithm to decrease the network congestion probability, which also helped to reduce bandwidth consumption [29].

Power consumption and traffic reduction are both considered. A topology-aware VM placement algorithm was developed, where groups of VMs are preferably placed in small areas of a datacenter network [30]. This algorithm took the power consumption of PMs and switches into account and switched off idle PMs. Yang *et al.* formulated a joint optimization problem for datacenters considering both energy consumption and communication load [31]. The authors put forward a job-aware VMP and route scheduling scheme to keep a datacenter's power consumption as low as possible while meeting quality-of-service requirements as much as possible. Chakravarthy *et al*. studied the problem of jointly optimizing VM scheduling and routing and formulated it as an integer programming problem [32]. A phase-wise metaheuristic algorithm was proposed to tackle the problem, considering the topology features and communication patterns. Gopu *et al*. modeled a multiobjective VMP problem in a distributed cloud, minimizing the three objectives of

wastage, power consumption, and propagation delay [33]. The multiobjective evolutionary algorithm based on decomposition (MOEA/D) was adapted to obtain high-quality nondominated solutions.

On the other hand, ACOs have been successfully applied to VMP problems. Alharbi *et al*. formulated the least energy consumption VMP problem as a constrained combinatorial optimization problem [18]. An ant colony system (ACS) with problem-specific heuristics was developed to obtain an energy-efficient solution. These heuristics are based on PM usage and capacity and thus prefer active PMs rather than switching on new PMs. Zhao *et al*. coped with the high power consumption and VM performance degradation in VMP. A power-aware and performance-guaranteed multiobjective ACO was proposed to optimize the two objectives above simultaneously [19]. Liu *et al.* proposed an ACO for the VMP problem, ACO-VMP, to minimize the number of active physical servers [34]. The pheromone was deposited between every two VMs, and the heuristic information between VM and PM indicates how the resource utilization is improved. Later on, the authors proposed an ACS algorithm incorporating order exchange and local search techniques that reduce the number of active servers in both homogeneous and heterogeneous environments [35]. Ants were guided by the pheromone deposited before to explore promising solutions via VM grouping. Wei *et al*. developed an adaptive multiobjective ACO to reduce the power consumption and communication cost in traffic-aware datacenter networks [36].

In summary, the existing VMP problems have been concerned with the power consumption of servers and network devices, communication traffic forwarding, network resource utilization, and so on. However, a little research has been conducted on jointly optimizing the power consumption of servers/switches and the network bandwidth resource consumption. Table 1 shows the comparison of all VMP algorithms reviewed in this section regarding the technique used, objective(s), advantages, and disadvantages. On the other hand, the above ACOs have shown excellent optimization performance when handling various VMP problems. ACOs in [18], [34], and [35] are for single objective VMP, while those in [19] and [36] are for bi-objective VMP, demonstrating their potential to address the new VMP problem formulated in this paper. Some of them introduced problem-specific knowledge to enhance the stochastic search procedure. They did, however, not pay enough attention to effective PM selection, VM ordering, and solution information exchange. It motivated us to adapt ACO for the VMP problem, with these three aspects considered.

Table 1 Comparison of the existing VMP algorithms

Refs	Technique used	Objective(s)	Advantages	Weaknesses
[20]	Greedy algorithm	• PC of Network	• Low traffic load compared with the random grouping	• High computational cost for frequent re-starting
[28]	Greedy algorithm	• System load	• Good robustness and adaptivity	• Computational feasibility is not ensured
[29]	Greedy algorithm	• Network traffic	• High feasibility for various priority and QoS requirements	• Only coarse-grained priority is considered
[21]	Heuristic	• PC of PMs • Service and security	• Provide a tradeoff between objectives	• Higher computation complexity than bin-packing heuristics
[25]	Heuristic	• Network traffic	• Design a metric "satisfactory" to reflect VM performance	• CPU and memory resources are ignored
[27]	Heuristic	• Bandwidth utilization	• Predictable latency performance	• Low feasibility if the time period is extended
[30]	Heuristic	• PC of PMs and switches	• Good scalability • High accessibility	• Network load balancing is ignored
[31]	Heuristic	• PC of PMs and switches • Communication load	• Load balancing	• Poor scalability for lightweight task dominated workload
[22]	Linear regression	• CPU and memory usage • VM migration	• Identify causes of SLA violations • Predict resource utilization	• Poor scalability • Network resource is ignored
[24]	Linear regression	• PC of PMs • Number of active PMs	• Identify overloaded and underloaded servers	• Impact of network load is ignored
[23]	Stochastic algorithm	• PC of PMs	• High feasibility	• Network resource is ignored
[26]	Approximation	• Connection cost	• Develop centralized and distributed models	• Bandwidth constraints and dynamic requests are ignored
[33]	MOEA/D	• PC of PMs • CPU wastage	• Low complexity	• Poor scalability • Bandwidth resource is ignored
[18]	ACS	• PC of PMs	• Good scalability	• High complexity
[19]	ACO	• PC of PMs • VM performance	• Good tradeoff between power consumption and VM performance	• Low feasibility for large-scale scenarios • Network resource is ignored
[32]	ACO	• PC of PMs and switches	• Good scalability	• PC of communication is ignored
[34]	ACO	• Number of active PMs	• Low complexity	• Network resource is ignored
[35]	ACO	• Number of active PMs	• Strong global exploration • High stability	• High complexity • Network resource is ignored
[36]	ACO	• PC of PMs and switches • Communication cost	• Fast convergence • Robust search capability	• High complexity

Note: PC is the abbreviation of power consumption.

2.2 Problem Description

In this paper, a datacenter is built based on the commonly used *Fat-Tree* topology, as illustrated in Fig. 1 [37]. Assume each switch owns M ports (M = 4 in Fig. 1). Then, there are M pods, and each pod contains $M/2$ aggregation switches, $M/2$ edge switches, and $M^2/4$ PMs. Besides, each pod is connected to $M^2/4$ core switches. A set of heterogeneous general-purpose PMs, i.e., servers, is maintained to provide cloud computing services to the public via software virtualization technologies. These services are running on virtual machines (VMs), which are instantiated dynamically and released on demand.

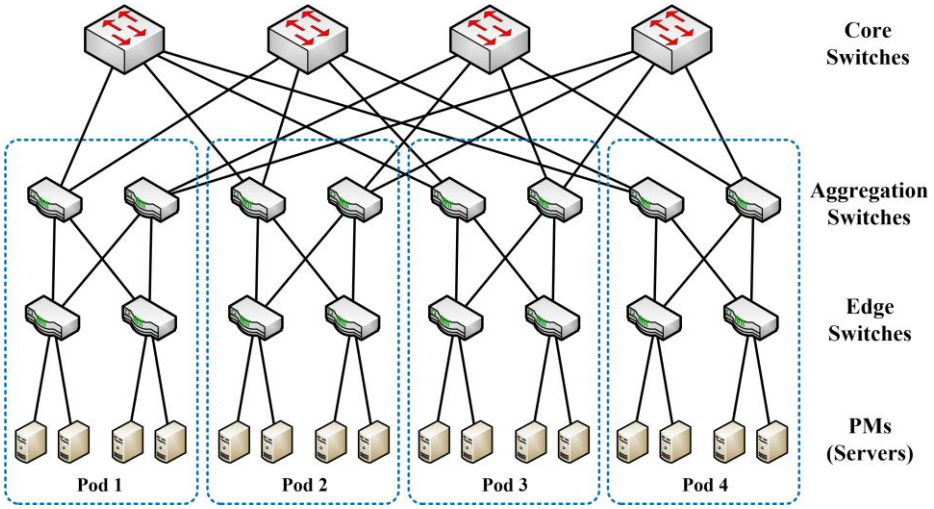


Fig. 1 An example Fat-Tree topology with four pods

To formulate the new VMP problem considered in this paper, we define the following notations.

Notation	Definition
E	The set of links in the datacenter.
e_i	The i-th link in E, $e_i \in E$.
λ	The link load coefficient that controls the maximum available bandwidth on each link. If λ = 1, the maximum available bandwidth on link e is equal to the maximum bandwidth that e can offer.
V	The set of VMs for placement, $V = \{v_1,\ldots,v_{\|V\|}\}$, where $\|V\|$ is the cardinality of V.
v_i	The i-th VM in V, $v_i \in V$.
P	The set of available PMs for hosting VMs in V, $P = \{p_1,\ldots,p_{\|P\|}\}$, where $\|P\|$ is the cardinality of P.
p_j	The j-th PM in P, $p_j \in P$.
V_{p_j}	The set of VMs that are hosted by PM p_j, $p_j \in P$ and $V_{p_j} \subseteq V$.
$p(v_i)$	The PM that hosts VM v_i, $p(v_i) \in P$.
y_{p_j}	A binary variable indicating if PM p_j is switched on or off. $y_{p_j} = 1$ if p_j is switched on, and $y_{p_k} = 0$, otherwise.
$R_{v_i}^{cpu}$	The CPU resource needed for hosting VM v_i.
$R_{v_i}^{mem}$	The memory resource needed for hosting VM v_i.
$C_{p_j}^{cpu}$	The CPU capacity of PM p_j.
$C_{p_j}^{mem}$	The memory capacity of PM p_j.
$U_{p_j}^{cpu}$	The already occupied CPU resource of PM p_j, $U_{p_j}^{cpu} \le C_{p_j}^{cpu}$.
$U_{p_j}^{mem}$	The already occupied memory resource of PM p_j, $U_{p_j}^{mem} \le C_{p_j}^{mem}$.
$u_{p_j}^{cpu}$	The CPU utilization ratio of PM p_j, i.e. ratio of $U_{p_j}^{cpu}$ to $C_{p_j}^{cpu}$, $0 \le u_{p_j}^{cpu} \le 1$.

T_{v_i,v_j}	The traffic amount between VMs, v_i and v_j.
N_{p_i,p_j}	The number of switches along the communication path between PMs, p_i and p_j. If the two PMs are connected by the same edge switch, $N_{p_i,p_j}=1$; if they belong to the same pod and connected to different edge switches, $N_{p_i,p_j}=3$; if their communication involves a core switch, then $N_{p_i,p_j}=5$. Details can be seen in Fig. 1.
$B_{e_i}^{max}$	The maximum bandwidth of link e_i.
$B_{e_i}^{csmd}$	The already consumed bandwidth of link e_i, $B_{e_i}^{csmd} \le B_{e_i}^{max}$.
$\gamma_{v_i,v_j}^{e_i}$	A binary variable indicating whether link e_i is used to support the communication between VMs, v_i and v_j. $\gamma_{v_i,v_j}^{e_i}=1$ means e_i is employed and $\gamma_{v_i,v_j}^{e_i}=0$, otherwise.
$Type^{switch}$	The set of switch types in the datacenter, $Type^{switch}=\{type_1,...,type_{\lvert Type^{switch}\rvert}\}$, and $\lvert Type^{switch}\rvert$ is the cardinality of $Type^{switch}$. In this paper, two types of switches are used, namely $type_{core}$ and $type_{agg,edge}$. $type_{core}$ is for core-level switching while $type_{agg,edge}$ is for aggregation/edge-level switching.
N_k^{switch}	The number of type-k switches, $k \in Type^{switch}$.
N_k^{ports}	The number of ports in a type-k switch, $k \in Type^{switch}$.
s_i^k	The i-th type-k switch, $i = 1,\ldots,$ N_k^{switch} and $k \in Type^{switch}$.
$Rate^{port}$	The set of the data rate modes supported by a switch port, where $\lvert Rate^{port}\rvert$ is the cardinality of $Rate^{port}$. In this paper, three data rate modes are supported by an arbitrary switch port, where $Rate^{port} = \{10, 100, 1000\}$ Mb/s.
$rate$	The maximum data rate at which a switch port forwards data, $rate \in Rate^{port}$. It is noted that for an arbitrary switch port, a larger $rate$ leads to a higher power consumption.
$Port_{k,i,j}^{rate}$	The j-th port of the i-th type-k switch working at a maximum data rate of $rate$ Mb/s.
$Power(Port_{k,i,j}^{rate})$	The amount of power consumed by switch port $Port_{k,i,j}^{rate}$.
$Power(idle, p_j)$	The power consumption of PM p_j when it is idle.
$Power(busy, p_j)$	The power consumption of PM p_j when it is running at its full capacity.
$Power(actual, p_j)$	The actual power consumption of PM p_j.
σ_{p_j}	The ratio of $Power(idle, p_j)$ to $Power(busy, p_j)$, where σ_{p_j} is set to 0.7 in this paper.
$Power(actual, s_i^k)$	The actual power consumption of switch s_i^k.
$Power(chassis, s_i^k)$	The amount of power consumed by the chassis of switch s_i^k.
$Power$(PMs)	The total amount of power consumed by all PMs.
$Power$(Switches)	The total amount of power consumed by all switches.
$Power_{total}$	The total amount of power consumed by all PMs and switches.
R_{total}^B	The total amount of bandwidth resource consumed by the communication among all VMs.

Assume a series of VM placement (VMP) tasks consistently arrive at the datacenter. Once a VMP task arrives, the datacenter selects a subset of PMs from $P = \{p_1,\ldots,p_{|P|}\}$ to host a given set of VMs, $V = \{v_1,\ldots,v_{|V|}\}$ [38]. This paper is concerned with two important issues to facilitate efficient operations at a datacenter, namely the power and bandwidth resource consumption. If no VMP task arrives, all idle PMs can be switched off for energy-saving purposes.

A. Power Consumption

Deploying a VM consumes CPU and memory resources on the corresponding PM. Assume that the power consumption is in proportion to the CPU utilization ratio on that PM [39]. A higher CPU utilization ratio leads to a higher amount of power consumption. When a PM is busy (i.e., running at its full capacity), the maximum power consumption incurs on that machine. However, when a PM is idle, its actual power consumption still accounts for 50~70 % of its maximum power consumption [40]. Hence, for a datacenter, its power consumption is related to how VMs are placed. In this paper, two types of general-purpose PMs are considered (see Subsection 5.1 for details).

A datacenter's power consumption mainly comes from active PMs, cooling systems, and those network devices forwarding data flows among PMs. However, as cooling systems are relatively isolated, where to place VMs does not affect the power consumption for cooling much. Hence, the power consumed by cooling is not considered in this paper.

In the Fat-Tree topology, switches are the most energy-consuming network devices (see Fig. 1). For simplicity, we consider switches as the only network devices that cause power consumption. For an arbitrary switch, its power consumption mainly comes from the base chassis and ports. The power consumed by a base chassis and that by a port depend on the switch type and the data rate mode being used, respectively. Two types of switches are considered in this paper, one for core-level switching and the other for aggregation/edge-level switching. At any time, a switch port is associated with a data rate mode, e.g., 10 Mb/s, 100Mb/s, or 1000Mb/s. Different data rate modes result in different power consumption. So, even for switches of the same type, if they employ different data rate modes in their ports, the actual power consumption is different [41][42]. In this paper, the total amount of power consumed by a datacenter, $Power_{total}$, is composed of the power consumption of all PMs and all switches, as defined in Eq. (1).

$$Power_{total} = Power(\text{PMs}) + Power(\text{Switches}) \tag{1}$$

The total power consumption of all PMs, $Power(\text{PMs})$, is defined in Eq. (2).

$$Power(\text{PMs}) = \sum_{j=1}^{|P|} Power(actual, p_j) \tag{2}$$

The power consumption of PM p_j, $Power(actual, p_j)$, is defined in Eq. (3). If p_j does not host any VM, it is switched off for energy-saving purpose and y_{p_j} is set to 0; otherwise, y_{p_j} is set to 1. If p_j is idle, its power

consumption is fixed as $Power(idle, p_j)$ and $u_{p_j}^{cpu}$ is set to 0; if p_j is hosting VMs, the corresponding power consumption is in proportion to $u_{p_j}^{cpu}$.

$$Power(actual, p_j) = y_{p_j} \cdot \left(Power(idle, p_j) + (1 - \sigma_{p_j}) \cdot Power(busy, p_j) \cdot u_{p_j}^{cpu} \right) \tag{3}$$

The total power consumed by all switches, *Power*(Switches), is defined in Eq. (4), where two types of switches are considered, i.e., $Type^{switch} = \{type_{core}, type_{agg,edge}\}$.

$$Power(\text{Switches}) = \sum_{k=1}^{|Type^{switch}|} \sum_{i=1}^{N_k^{switch}} Power(actual, s_i^k) \tag{4}$$

Eq. (5) defines the power consumption of the *i*-th type-*k* switch, where $i = 1,\ldots, N_k^{switch}$, and $k \in Type^{switch}$. For an arbitrary switch port, its power consumption depends on the data rate mode this port is working at. A port with a higher data rate leads to higher power consumption.

$$Power(actual, s_i^k) = Power(chassis, s_i^k) + \sum_{j=1}^{N_k^{ports}} Power(Port_{k,i,j}^{rate}) \tag{5}$$

B. Bandwidth Resource

Data exchange among VMs is fulfilled by communication among PMs hosting them. If VMs are not properly placed, links that support the communication among these VMs may become overloaded. That could cause larger end-to-end delays and even network congestion, leading to severe performance deterioration. Therefore, when minimizing the power consumption of a datacenter, we should also consider network performance.

Bandwidth resource consumption is one of the key issues when evaluating network performance [43], as it is, to a certain extent, an indicator of how much link resource is consumed. For two arbitrary VMs, the bandwidth resource consumption is dependent on the locations of their hosts and the traffic demand between them. If the two VMs are deployed on the same PM, no bandwidth resource consumption results; otherwise, the bandwidth resource consumption incurs along the communication path between the two PMs hosting the two VMs. When a VMP task arrives, it specifies the traffic demand between any pair of VMs. In other words, the traffic demand between any two VMs is predetermined by that task. Given an arbitrary pair of VMs, the datacenter only delivers the predetermined amount of traffic between them as requested. There is more than one shortest path between each pair of PMs because of the Fat-Tree topology. Hence, we can select the most appropriate path for traffic delivery with respect to criteria, such as power consumption and bandwidth utilization. Provided that the traffic demand between the VMs is fixed, the bandwidth resource

consumption is proportional to the number of switches along the communication path between the corresponding PMs. We define the total bandwidth resource consumed by all VMs in Eq. (6) [44].

$$R_{total}^{B} = \sum_{v_i \in V} \sum_{v_j \in V} N_{p(v_i), p(v_j)} \cdot T_{v_i, v_j} \tag{6}$$

where, $N_{p(v_i), p(v_j)}$ denotes the number of switches along the communication path between two PMs, $p(v_i)$ and $p(v_j)$. T_{v_i, v_j} is the traffic amount between two VMs, v_i and v_j.

C. The VM Placement Problem

Once a VMP task arrives, the VMP problem aims to jointly minimize the power consumption of all PMs and switches and the bandwidth resource consumption among VMs, satisfying all resource constraints. We aggregate the two objectives into a single cost function, as defined in Eq. (7), where θ_1 and θ_2 are two weights indicating the relative importance of $Power_{total}$ and R_{total}^{B}, respectively. θ_1 and θ_2 are regarded as prior knowledge. This paper assumes that decision-maker has prior knowledge and hence can set θ_1 and θ_2 properly. In this paper, both θ_1 and θ_2 are set to 0.5. The VMP problem is defined to minimize the cost function.

Minimize:

$$Cost = \sqrt{\theta_1 \cdot (Power_{total})^2 + \theta_2 \cdot (R_{total}^{B})^2} \tag{7}$$

Subject to:

$$\theta_1 + \theta_2 = 1, \quad \forall 0 \le \theta_1, \theta_2 \le 1 \tag{8}$$

$$\bigcup_{p_j \in P} V_{p_j} = V \tag{9}$$

$$V_{p_i} \bigcap V_{p_j} = \varnothing, \quad \forall p_i, p_j \in P, p_i \neq p_j \tag{10}$$

$$\sum_{v_i \in V_{p_j}} R_{v_i}^{cpu} \le C_{p_j}^{cpu} \tag{11}$$

$$\sum_{v_i \in V_{p_j}} R_{v_i}^{mem} \le C_{p_j}^{mem} \tag{12}$$

$$B_{e_i}^{csmd} = \sum_{v_i \in V} \sum_{v_j \in V} \gamma_{v_i, v_j}^{e_i} \cdot T_{v_i, v_j} \le \lambda B_{e_i}^{max} \tag{13}$$

Constraint (8) defines that the summation of θ_1 and θ_2 is equal to 1, where θ_1 and θ_2 are in the range of [0,1], respectively. Constraint (9) defines that all VMs in V must be successfully placed. Constraint (10) ensures that each VM is placed on a single PM. Constraints (11)-(12) define that the CPU and memory resources occupied by all VMs on PM p_j cannot exceed the maximum CPU and memory capacities of p_j, respectively. Constraint (13) defines that the bandwidth

consumption of link e_i cannot exceed a threshold value, $\lambda B_{e_i}^{max}$, where λ is the link load coefficient not larger than 1. This constraint ensures all physical links are not overloaded.

3. ACO for Virtual Machine Placement

When searching for food, ants deposit a chemical substance on their paths, i.e., pheromone. Other ants are then more likely to choose those paths with a higher amount of pheromone. Over time the pheromone trails in promising paths are gradually strengthened and reinforced with increasingly more ants passing. ACO is a population-based metaheuristic mimicking the foraging behavior of ants [13]. Pheromone is adopted to record the historical information accumulated by ants during the construction of previous solutions. Besides, heuristic information is used to guide the search for promising solutions in a greedy manner.

A number of ACOs have been developed to address the VMP problem [18][19][34][35]. The 2-dimension binary encoding is commonly used for solution representation, where each element in a matrix, e.g., X_{ij}, represents if VM v_i is to be placed on PM p_j. X_{ij} = 1 means p_j is selected to host v_i and X_{ij} = 0, otherwise. For an arbitrary ant, the number of moves to construct a solution is equal to that of VMs. At each move, the ant chooses a PM to deploy a particular VM. Fig. 2 illustrates an example of the solution construction process, where four VMs are to be placed according to the pheromone trails and the heuristic information. The trajectory (bolded arrows) shows a solution to the problem, where VM_1, VM_2, VM_3, and VM_4 are placed on PM_3, PM_1, PM_2, and PM_4, respectively.

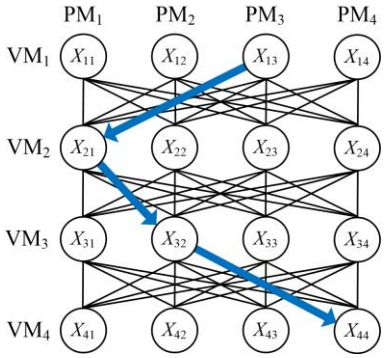


Fig. 2 An example of the solution construction process in ACO

In ACO, both pheromone and heuristic information is used to construct solutions. Pheromone $\tau(v_i, p_j)$ is deposited between each VM-PM pair, $v_i \in V$, $p_j \in P$. Heuristic information $\eta(v_i, p_j)$ provides problem specific knowledge indicating how desirable v_i is to be installed on p_j. An ant constructs a solution by iteratively selecting a PM $p \in P$ to place a VM $v_i \in V$ according to the pseudo-random proportional rule in Eq. (14).

$$p = \begin{cases} \arg\max_{p_j \in serverList(v_i)} \{\tau(v_i, p_j)^{\alpha} \cdot \eta(v_i, p_j)^{\beta}\}, & q \le q_0 \\ p_{RND}, & otherwise \end{cases} \quad (14)$$

where, $serverList(v_i)$ is the set of available PMs for hosting v_i; q is randomly generated in the range of [0,1]; q_0 is a constant between 0 and 1 that reflects the relative importance of exploiting the already accumulated domain knowledge against exploring new moves; α and β are real numbers reflecting the relative importance of pheromone trail and heuristic information, respectively. If $q \le q_0$, exploitation is implemented, and the PM with the largest $\tau(v_i, p_j)^{\alpha} \cdot \eta(v_i, p_j)^{\beta}$ value is selected to host v_i; otherwise, exploration is carried out, and the roulette wheel selection generates a random PM p_{RND} from $serverList(v_i)$ to host v_i. Eq. (15) defines the probability $prob(v_i, p_j)$ of deploying v_i on p_j. In the roulette wheel selection, the j-th PM in $serverList(v_i)$ is selected with probability $prob(v_i, p_j)$.

$$prob(v_i.p_j) = \frac{\tau(v_i, p_j)^{\alpha} \cdot \eta(v_i, p_j)^{\beta}}{\sum_{p_k \in serverList(v_i)} \tau(v_i, p_k)^{\alpha} \cdot \eta(v_i, p_k)^{\beta}} \quad (15)$$

4. The Proposed ACO

This section first introduces the solution encoding. After that, the three novel performance-enhancing schemes are described in detail, including the energy- and bandwidth-aware physical machine selection (EBAPMS) scheme, the traffic-based virtual machine ordering (TVMO) scheme, and the direct information exchange (DIEX) scheme. In the end, we present the overall procedure of the proposed energy- and traffic-aware ACO (ETA-ACO).

4.1 Solution Encoding and Evaluation

In the literature, 2-dimensional binary encoding is usually used to represent VMP solutions (see Fig. 2 as an example). To reduce space complexity, we convert the 2-dimensional binary encoding to a 1-dimensional integer encoding. Each PM is associated with an integer ID. An integer string consists of a number of PM IDs, reflecting how a set of VMs is to be hosted, i.e., the integer ID at each position indicates on which PM the corresponding VM is placed. Integer encoding simplifies the solution representation and is suitable for the direct information exchange scheme (see Subsection 4.4 for details). Fig. 3 shows an example of how integer encoding represents the solution in Fig. 2.

VM_1	VM_2	VM_3	VM_4
3	1	2	4

Fig. 3 An example of the integer-encoding-based solution

4.2 Energy- and Bandwidth-aware PM Selection Scheme

In the existing ACOs for VMP, the heuristic information, $\eta(v_i, p_j)$, is usually defined according to either the power consumption or the consumed power and network resources [18][19][34][35][36]. In the first case, power consumption is the only objective to minimize while network bandwidth resource is not considered. A datacenter may result in minimum power consumption at the expense of a significant bandwidth resource consumption, leading to severe link overload, increased communication delay, and even network congestion. Therefore, when tackling energy-aware VMP problems, we should not ignore the bandwidth resource consumption. In the second case, the heuristic information relies on power and network consumption. The PM for hosting v_i is selected according to the corresponding heuristic value. In this case, it looks like the power and the bandwidth resource consumption is minimized simultaneously. However, due to the different metrics used, e.g., kW·h and Mb/s, it is hard to unify their dimensions properly. It is not easy to strike a balance between power and bandwidth resource consumption.

To address the above issue, an energy- and bandwidth-aware PM selection (EBAPMS) scheme is proposed based on two types of values, namely, the heuristic information and the product coefficient value as follows.

The heuristic information, $\eta(v_i, p_j)$, is updated according to Eq. (16), where n is the number of VMs already placed. $\eta(v_i, p_j)$ consists of two components, $\eta_1(v_i, p_j)$ and $\eta_2(v_i, p_j)$, reflecting how CPU and memory resources of PMs are balanced and how much power is consumed, respectively.

$$\eta(v_i, p_j) = \left(1 + \frac{n}{|V|}\right) \cdot \eta_1(v_i, p_j) + \left(1 - \frac{n}{|V|}\right) \eta_2(v_i, p_j) \tag{16}$$

$\eta_1(v_i, p_j)$ is defined in Eq. (17). A larger $\eta_1(v_i, p_j)$ means a more balanced trade-off between CPU and Memory resources is achieved if v_i is placed on p_j. It helps to accommodate more VMs.

$$\eta_1(v_i, p_j) = \left(\frac{|(C_{p_j}^{cpu} - U_{p_j}^{cpu} - R_{v_i}^{cpu}) - (C_{p_j}^{mem} - U_{p_j}^{mem} - R_{v_i}^{mem})|}{U_{p_j}^{cpu} + R_{v_i}^{cpu} + U_{p_j}^{mem} + R_{v_i}^{mem}} + \sum_{k=1, k\neq j}^{|P|} \frac{|(C_{p_k}^{cpu} - U_{p_k}^{cpu}) - (C_{p_k}^{mem} - U_{p_k}^{mem})|}{U_{p_k}^{cpu} + U_{p_k}^{mem}} \right)^{-1} \tag{17}$$

$\eta_2(v_i, p_j)$ is defined in Eq. (18), where $Power(total, p_k)$ is the actual power consumption of p_k and $Power'(total, p_j)$ represents the actual power consumption of p_j if v_i is hosted by p_j. If $y_{p_k} = 1$, it means p_k is switched on and $y_{p_k} = 0$, otherwise. A larger $\eta_2(v_i, p_j)$ means the total power consumption of all PMs is lower.

$$\eta_2(v_i, p_j) = \left(\sum_{k=1, k\neq j}^{|P|} y_{p_k} \right) \cdot \left(Power'(actual, p_j) + \sum_{k=1, k\neq j}^{|P|} Power(actual, p_k) \right)^{-1} \tag{18}$$

When considering the placement of v_i, we maintain a server list containing all available PMs for hosting v_i,

serverList(v_i). For an arbitrary PM $p_j \in serverList(v_i)$, a product coefficient value, Γ_{v_i,p_j}, is defined as the product of $\tau(v_i, p_j)$ and $\eta(v_i, p_j)$ in Eq. (19).

$$\Gamma_{v_i,p_j} = \tau(v_i, p_j)^{\alpha} \cdot \eta(v_i, p_j)^{\beta} \tag{19}$$

The EBAPMS scheme prioritizes the minimization of power consumption while reducing the bandwidth resource consumption as much as possible. It helps to achieve a lower bandwidth cost without sacrificing too much power. The scheme is introduced as follows.

- **Step 1**: for each (v_i, p_j) pair, its heuristic information $\eta(v_i, p_j)$ is calculated. For each v_i, the product coefficient values $\Gamma_{v_i,p_j}, p_j \in serverList(v_i)$ are calculated, and the largest one is found, i.e., $\Gamma^{\max}_{v_i,p_j} = \max\{\Gamma_{v_i,p_j} \mid p_j \in serverList(v_i)\}$.
- **Step 2**: those PMs with a product coefficient value smaller than $0.7 \times \Gamma^{\max}_{v_i,p_j}$ are removed from *serverList*(v_i). The remaining PMs in *serverList*(v_i) ensure the solutions are of high quality regarding the power consumption.
- **Step 3**: the product coefficient values Γ_{v_i,p_j} of those remaining $p_j \in serverList(v_i)$ are updated by using Eq. (20). The traffic impact is added to the product coefficient for bandwidth resource reduction. The results are then used to determine the PM for hosting v_i, according to the pseudo-random proportional rule in Section 3.

$$\Gamma_{v_i,p_j} = \Gamma_{v_i,p_j} \cdot \left(1 + \sum_{v_k \in V_{p_j}} TC_{v_i,v_k}\right) \tag{20}$$

where, TC_{v_i,v_k} is the ratio of the traffic demand between v_i and v_k to the traffic demand among all VMs.

4.3 Traffic-based VM Ordering Scheme

If two VMs are placed on the same PM, the communication between them does not consume network bandwidth resources. If the two VMs are running on two different PMs, the bandwidth resource consumption depends on where the two PMs are located. Note that network link resources can be converted to bandwidth resources. Given two VMs, even though their traffic demand is fixed, more switches are on the path between them consumes more bandwidth resources. That is because more links are involved in packet exchange. For example, VMs belonging to the same pod and those in different pods lead to different network bandwidth consumption. If we place two VMs on two PMs far from each other, more bandwidth resources are consumed as more switches are involved in relaying the communication. If VMs are

placed randomly, VM pairs with higher traffic demand may be deployed on PMs far-away from each other, leading to severe bandwidth resource consumption. Hence, it is rational to place those pairs of VMs with higher traffic demand on PMs as close as possible, reducing the total bandwidth resource consumption. The ordering of placing VMs has not received sufficient attention in the literature [18][19][35].

Inspired by the above, a traffic-based VM ordering (TVMO) scheme is designed to place those VM pairs with higher traffic demand earlier than others. Thus, they are more likely to be placed on closer PMs or even the same PM. It is no doubt that the VM ordering scheme helps, to a certain extent, to save the total bandwidth resource consumption without incuring much computational costs. To be specific, all VMs are stored in a VM list, *vmList*, in descending order regarding their traffic demand. Let $\boldsymbol{T}_{traffic}$ be the traffic table demanded by VMs, where each row (v_i, v_j, T_{v_i,v_j}) records the traffic demand between every pair of VMs. Let $|\boldsymbol{T}_{traffic}|$ denote the number of VM pairs in $\boldsymbol{T}_{traffic}$. Fig. 4 shows the procedure to generate *vmList*.

Initialization:

1. Set *vmList* = ø , $\boldsymbol{T}_{temp} = \boldsymbol{T}_{traffic}$, and $k = 1$;
2. Sort the rows in $\boldsymbol{T}_{temp}$ in descending order by the traffic demand T_{v_i,v_j};

Repeat:

3. Pick up pair (v_i, v_j) from the k-th row in $\boldsymbol{T}_{traffic}$;
4. **if** v_i *is not in vmList* **then** Add v_i to *vmList*;
5. **if** v_j *is not in vmList* **then** Add v_j to *vmList*;
6. Set $k = k + 1$;

Until $k > |\boldsymbol{T}_{traffic}|$

7. Output *vmList* as the VM placement list

Fig. 4 Procedure of generating the ordered VM list *vmList*.

4.4 The Direct Information Exchange Scheme

In traditional ACOs, information exchange among ants is realized by pheromone trails, where ants choose paths with higher pheromone levels. Such information exchange is indirect, as illustrated in Fig. 2. Unlike the indirect information exchange in ant colony, human-beings exchange information by faster and direct communication. The above motivated us to introduce a direct information exchange (DIEX) scheme.

Assume $\boldsymbol{Place}(k)=(p^k(v_1),...,p^k(v_{|V|}))$ is the solution constructed by ant k, where $p^k(v_i)$ is the PM for hosting v_i. Power consumption $Power_{\boldsymbol{Place}(k)}$ is calculated by using Eq. (21). Let N_{ants} be the number of ants in ETA-ACO. In Eq. (21),

Power(***Place***(*k*), PMs) and *Power*(***Place***(*k*), Switches) are the power consumption of all PMs (see Eq. (2)) and that of all switches (see Eq. (4)) based on ***Place***(*k*), respectively.

$$Power_{\boldsymbol{Place}(k)} = Power\left(\boldsymbol{Place}(k), \text{PMs}\right) + Power\left(\boldsymbol{Place}(k), \text{Switches}\right) \tag{21}$$

```
// Find the minimum power consumption Power_min
for k = 1 to N_ants do
    Evaluate Place(k) and obtain Power_Place(k) according to Eq. (20);
Find the minimum power consumption Power_min and its corresponding ant, e.g. ant h;
Set Place_best = Place(h) and Power_best = Power_min;
// Construction of Place'(k)
for k = 1 to N_ants do
    if k ≠ h then do
        Set Place'(k) = Place(k);
        for i = 1 to |V| do
            Generate a random number RND between 0 and 1;
            if RND < Power_Place(k) / (Power_Place(k) + Power_Place(h)) then do
                Set p'^k_new(v_i) = p^h(v_i);
        Evaluate Place'(k) and obtain Power_Place'(k);
// Update of Place'(k) and Place_best
        if Place'(k) is feasible then do
            if Power_Place'(k) ≤ Power_Place(h) then do
                Replace Place(h) with Place'(k);
            if Power_Place'(k) ≤ Power_best then do
                Replace Place_best with Place'(k);
                Replace Power_best with Power_Place'(k);
```

Fig. 5 Procedure of the DIEX scheme

In the DIEX scheme, the ant with the minimum power consumption, $Power_{min} = \min\{Power_{\boldsymbol{Place}(k)} | k = 1, \ldots, N_{ants}\}$, is used to exchange solution information with the other ants in the colony. Assume ant h has the minimum power consumption, i.e., $Power_{\boldsymbol{Place}(h)} = Power_{min}$. Suppose ant k is the one to exchange solution information with ant h. A new solution $\boldsymbol{Place}'(k)=(p^k_{new}(v_1),..., p^k_{new}(v_{|V|}))$ is constructed for ant k. If ***Place***'(*k*) is better than ***Place***(*k*), ***Place***(*k*) is replaced with ***Place***'(*k*). If ***Place***'(*k*) is better than the global best solution $\boldsymbol{Place}_{best}$, $\boldsymbol{Place}_{best}$ and its power consumption $Power_{best}$ are also updated. Thus, excellent solution components rapidly spread over the ant colony, guiding the search towards promising areas in the search space. As observed in Section 5, the DIEX scheme accelerates the solution

construction process and approaches promising solutions with a higher probability. Fig. 5 shows the procedure of the DIEX scheme.

4.5 The Procedure of the Energy- and Traffic-aware ACO

The proposed ETA-ACO is featured with three performance-enhancing strategies, including the EBAPMS scheme, the TVMO scheme, and the DIEX scheme. Fig. 6 shows the procedure of the proposed ETA-ACO algorithm.

Initialization:

1. Set generation $GEN = 0$;
2. Implement the TVMO scheme to sort VMs and place them to *vmList*; // see Subsection 4.3
3. For each VM-PM pair (v_i, p_j), set $\tau(v_i, p_j) = \tau_0$ and $\eta(v_i, p_j) = \eta_0$, where $i = 1,\ldots, |V|, j = 1,\ldots, |P|$;

Repeat:

4. Set $GEN = GEN + 1$;
5. **for** $k = 1$ **to** N_{ants} **do** // Solution construction of ant k
6. Initialize the occupied CPU and memory resources of each PM;
7. Initialize the occupied bandwidth on each link;
8. **for** $i = 1$ **to** $|V|$ **do** // placement of VM i
9. Obtain the set of available servers for hosting v_i, $serverList(v_i)$; // see Subsection 4.2
10. **for** $j = 1$ **to** $N_{serverList}$ **do** // for each PM in $serverList(v_i)$
11. Pick up PM j, p_j, from $serverList(v_i)$ and update the heuristic value $\eta(v_i,p_j)$;
12. Select a PM, e.g. p_j, from $serverList(v_i)$ according to the EBAPMS scheme; // see Subsection 4.2
13. Include the selected PM p_j as the one for placing v_i;
14. Update the occupied CPU and memory resources of each PM and the occupied bandwidth on each link;
15. Invoke the local pheromone updating process;
16. Evaluate all solutions constructed by the ant colony;
17. Implement the DIEX scheme and update $\boldsymbol{Place}_{best}$ and $Power_{best}$; // see Subsection 4.4
18. Invoke the global pheromone updating process;

Until $GEN = GEN_{max}$

19. Output $\boldsymbol{Place}_{best}$, and its corresponding $Power_{best}$ and R^B_{best};

Fig. 6 Overall procedure of the proposed ETA-ACO

In Step 15, $\tau(v_i, p_j)$ undergoes the local pheromone update process as defined in Eq. (22).

$$\tau(v_i, p_j) = (1-\rho_l)\cdot\tau(v_i, p_j) + \rho_l \cdot \tau_0 \tag{22}$$

In Step 18, the global pheromone update process is defined in Eq. (23), where $Power_{max}$ and $Power_{min}$ are the

maximum and minimum values of the total power consumption, respectively.

$$\tau(v_i, p_j) = (1-\rho_g)\cdot\tau(v_i, p_j) + \rho_g \cdot \frac{Power_{max}}{Power_{min}} \tag{23}$$

The termination condition is that a predefined number of generations is reached.

ETA-ACO was specifically devised for the VMP problem concerned in this paper but not for general optimization problems. So, ETA-ACO may not handle many-objective optimization problems that are usually tackled by multi-objective optimization algorithms such as NSGA-III.

4.6 Time Complexity Analysis

The procedure of ETA-ACO consists of the initialization and the main loop. In Step 2, the TVMO scheme sorts the VMs according to the communication demand. This results in a time complexity of $O(|V|\cdot log|V|)$, where $|V|$ is the number of VMs. In Step 3, the heuristic and pheromone values are initialized, respectively, leading to a complexity of $O(|V|\cdot|P|)$, where $|P|$ is the number of PMs. Hence, the initialization has a time complexity of $O(|V|\cdot log|V| + |V|\cdot|P|)$.

The main loop incurs the most time-consuming computational burden. Steps 5-15 construct N_{ants} solutions. Specifically, Steps 6-7 initialize the physical resources of PMs and the bandwidth resources of links, respectively, each requiring a time complexity of $O(|P|\cdot N_{ants})$. Steps 9-11 calculates the heuristic values between a VM and a number of PMs, which leads to $O(|V|\cdot|P|\cdot N_{ants})$. Step 12 is based on the EBAPMS scheme and its time complexity is $O(|V|\cdot|P|\cdot N_{ants})$. Steps 13-15 update the local pheromone and physical resource consumption of each VM-PM pair, which also requires a time complexity of $O(|V|\cdot|P|\cdot N_{ants})$. So, the time complexity of Steps 5-15 is $O(|P|\cdot N_{ants} + |V|\cdot|P|\cdot N_{ants})$, which can be reduced to $O(|V|\cdot|P|\cdot N_{ants})$. Denote the time complexity for evaluating a solution by $O(f)$. Step 16 requires $O(f\cdot N_{ants})$ to evaluates all N_{ants} solutions. Step 17 applies the DIEX scheme to the constructed solutions, which leads to a time complexity of $O(|V|\cdot N_{ants})$. Step 18 invokes the global pheromone updating process, resulting in $O(|V|)$. Hence, the time complexity of the main loop is expressed as $O((|V|\cdot|P|\cdot N_{ants} + f\cdot N_{ants} + |V|\cdot N_{ants} + |V|)\cdot GEN_{max}) = O((|V|\cdot|P| + f)\cdot N_{ants}\cdot GEN_{max})$.

The time complexity of ETA-ACO is thus $O(|V|\cdot log|V| + |V|\cdot|P| + N_{ants}\cdot GEN_{max}\cdot(|V|\cdot|P|+ f)) = O(|V|\cdot log|V| + N_{ants}\cdot GEN_{max}\cdot(|V|\cdot|P| + f))$. Compared with the main loop, the initialization is far less computationally expensive and thus can be ignored. Therefore, the time complexity of ETA-ACO is reduced to that of the main loop, i.e., $O((|V|\cdot|P| + f)\cdot N_{ants}\cdot GEN_{max})$.

On the other hand, a number of ACO algorithms have been proposed for VMP problems, e.g., [18][19][34][35][36].

The ACO in [18] has a time complexity of $O(N_k \cdot |P| \cdot ((|V| \cdot |P| + f) \cdot N_{ants} \cdot GEN_{max} + N_{slot\_int}))$, where N_k and N_{slot_int} are the numbers of intervals and slots in an interval, respectively. In [19], [34], and [36], the three ACOs lead to identical time complexity, namely $O((|V| \cdot |P| + f) \cdot N_{ants} \cdot GEN_{max})$. The ACO in [35] has a time complexity of $O((|V| \cdot |P|^2 + f) \cdot N_{ants} \cdot GEN_{max})$. That is because the ACO in [35] is extended from the ACO in [34] by introducing order exchange and migration (OEM) local search techniques, causing additional time complexity. It is easily seen that the proposed ETA-ACO has the same complexity as those in [19], [34], and [36]. Besides, its time complexity is lower than those in [18] and [35].

5. Performance Evaluation

This section first introduces test instances and performance metrics. It then evaluates the effectiveness of each performance-enhancing scheme. Finally, the proposed ETA-ACO is compared with a number of state-of-the-art heuristics and metaheuristics in terms of the overall optimization performance.

5.1 Test Instances

Since the problem concerned in this paper is a new variant of the VMP problem, existing benchmark instances are not immediately available. To thoroughly evaluate the performance of the proposed ETA-ACO, we generate 36 instances that are classified into four categories according to the number of PMs. The number of VMs to be placed, $|V|$, varies from 2 to 6 times of the number of PMs, $|P|$, mimicking different VM load pressures on a datacenter. For each ($|V|$, $|P|$) pair, three instances are randomly generated. For example, instances 1, 2 and 3 are generated with $|V| = 40$ and $|P| = 20$. Let $N_{k=core}^{switch}$, $N_{k=agg.}^{switch}$ and $N_{k=edge}^{switch}$ be the numbers of core-level, aggregation-level, and edge-level switches, respectively. Table 2 shows the 36 test instances and their parameters. This paper considers two types of flows, namely the elephant flow (E-flow) and the mouse flow (M-flow).

Table 2 Test instances and their parameters

Instances	$\|V\|$	$\|P\|$	$N_{k=core}^{switch}$	$N_{k=agg.}^{switch}$	$N_{k=edge}^{switch}$	$N_{cluster}^{E-flow}$	T_{v_i,v_j}^{E-flow}	$N_{cluster}^{M-flow}$	T_{v_i,v_j}^{M-flow}
I-1, I-2, I-3	40	20	9	18	18	5~10	25~100	3~10	1~5
I-4, I-5, I-6	60	20	9	18	18	5~10	25~100	3~10	1~5
I-7, I-8, I-9	120	20	9	18	18	5~10	25~100	3~10	1~3
I-10, I-11, I-12	80	40	9	18	18	5~15	25~100	5~10	1~5
I-13, I-14, I-15	120	40	9	18	18	5~15	25~100	5~10	1~5
I-16, I-17, I-18	240	40	9	18	18	5~15	25~100	5~10	1~3
I-19, I-20, I-21	120	60	16	32	32	10~15	25~100	5~15	1~5
I-22, I-23, I-24	240	60	16	32	32	10~15	25~100	5~15	1~5
I-25, I-26, I-27	360	60	16	32	32	10~15	25~100	5~15	1~3

I-28, I-29, I-30	160	80	16	32	32	15~20	25~100	10~15	1~5
I-31, I-32, I-33	240	80	16	32	32	15~20	25~100	10~15	1~5
I-34, I-35, I-36	480	80	16	32	32	15~20	25~100	10~15	1~3

In our experiments, VMs are randomly grouped into clusters, and communication incurs in every cluster. Each cluster is associated with a size and a flow-type. Without loss of generality, we assume that the number of E-flows is far less than that of M-flows, and there is only one E-flow cluster in each instance. Let $N_{cluster}^{E-flow}$ and $N_{cluster}^{M-flow}$ be the sizes of E-flow and M-flow clusters, respectively. $N_{cluster}^{E-flow}$ and $N_{cluster}^{M-flow}$ are randomly generated according to the ranges in Table 2. If VMs v_i and v_j belong to an E-flow cluster, the traffic demand between them, T_{v_i,v_j}^{E-flow}, is randomly generated according to the E-flow traffic range. Otherwise, T_{v_i,v_j}^{M-flow} is randomly generated according to the M-flow traffic range. T_{v_i,v_j}^{E-flow} and T_{v_i,v_j}^{M-flow} are measured by Mb/s.

Also, there are a number of instance-independent parameters. Two types of PMs are considered, namely Type-1 PM with 40000 MIPS CPU capacity and 32 GB memory and Type-2 PM associated with 24000 MIPS CPU capacity and 16 GB memory, respectively. If p_j is a Type-1 PM, $Power(busy, p_j)$ is set to 550 W; otherwise, $Power(busy, p_j)$ is set to 420 W. In this paper, the ratio of $Power(idle, p_j)$ to $Power(busy, p_j)$, σ_{p_j} is fixed to 0.7 [40][51]. A datacenter consists of 40% Type-1 and 60% Type-2 PMs. If s_i^k is a core switch, we have $Power(chassis, s_i^k)$ = 555 W; otherwise, we have $Power(chassis, s_i^k)$ = 150 W as it is either aggregation or edge switch. For a core switch, $Power(Port_{k,i,j}^{rate})$ with the maximum data rate *rate* = 10 Mb/s, 100 Mb/s and 1000 Mb/s, is set to 4 W, 8 W and 22 W, respectively, while for an aggregation/edge switch, $Power(Port_{k,i,j}^{rate})$ with *rate* = 10 Mb/s, 100 Mb/s and 1000 Mb/s, is set to 0.2 W, 0.4 W and 1.1 W, respectively. For an arbitrary link $e_i \in E$, we set the link load coefficient $\lambda = 0.8$ and the maximum bandwidth $B_{e_i}^{max}$ = 1000 Mb/s [27]. For an arbitrary VM v_i, its CPU and memory requirements, $R_{v_i}^{cpu}$ and $R_{v_i}^{mem}$, are randomly generated in the range of [1500, 5000] MIPS and [1.5, 5] GB, respectively. To encourage scientific comparison, test instances are available at http://www.cs.nott.ac.uk/~pszrq/benchmarks.htm. All experiments are run on a computer with Windows 10 OS, Intel Core i7-7500U CPU 2.7 GHz and 8 GB RAM. Our programming language is Java, and all codes are developed based on JDK 1.8.

5.2 Performance Metrics

To thoroughly evaluate ETA-ACO's performance, we adopt the following performance metrics throughout Section 5.

Each metaheuristic is run 30 times to collect 30 best solutions for statistical analysis. On the other hand, a number of deterministic heuristics are employed for performance comparison (see Subsection 5.4). Each of them is run once.

- The average objective function (*Cost*) value of the best solutions found over 30 runs.
- The average total power consumption ($Power_{total}$) value of the best solutions found over 30 runs.
- The average total bandwidth resource consumption (R_{total}^{B}) value of the best solutions found over 30 runs.
- The Friedman test, a non-parametric statistical test [46], for comparing multiple algorithms with respect to the cost values of the 30 best solutions. All algorithms for comparison are ranked, and their average ranks explicitly reveal how well they perform.
- The average computational time (ACT) consumed by an algorithm over 30 runs. ACT reflects the time complexity of an algorithm.

5.3 Effectiveness of the Three Schemes

Three novel schemes are developed to enhance the optimization performance of ETA-ACO, including the EBAPMS scheme in Subsection 4.2, the TVMO scheme in Subsection 4.3, and the DIEX scheme in Subsection 4.4. To demonstrate the effectiveness of each scheme, we compare the following four variants.

- A-1: The original ACO adapted for the VMP problem concerned in this paper.
- A-2: A-1 with the EBAPMS scheme.
- A-3: A-2 with the TVMO scheme.
- A-4: A-3 with the DIEX scheme, namely ETA-ACO.

Table 3 Mean values of *Cost*, $Power_{total}$ and R_{total}^{B} (Best results are in bold)

Instances	Average *Cost*				Average $Power_{total}$ (W)				Average R_{total}^{B} (Mb/s)			
	A-1	A-2	A-3	A-4	A-1	A-2	A-3	A-4	A-1	A-2	A-3	A-4
I-1	10189	10115	9608	**9291**	13807	13710	13479	**12995**	4124	4081	**1716**	1939
I-5	10808	10773	10419	**10105**	14745	14780	14709	**14268**	4028	3696	865	**808**
I-9	13800	13678	13252	**12826**	18647	18481	18521	**17957**	5759	5712	2868	**2560**
I-12	12547	12345	11861	**11458**	16854	16654	16551	**15936**	5548	5240	**2729**	2935
I-15	13903	13602	13125	**12755**	18823	18453	18387	**17885**	5683	5434	2535	**2349**
I-16	20315	20289	19341	**19133**	26799	26784	26692	**26420**	10356	10289	5974	**5840**
I-21	19283	19409	18832	**18049**	26825	26706	26544	**25435**	4911	6338	2166	**2139**
I-24	24481	24059	23322	**22650**	32995	32479	32363	**31443**	10486	10139	6364	**6111**
I-27	31132	31121	29944	**29209**	41829	41776	41587	**40548**	13736	13847	7989	**7883**
I-29	23022	22712	22118	**21252**	31353	30960	30633	**29379**	8774	8553	**6325**	6340
I-31	27452	27091	25877	**24607**	36765	36333	35941	**34154**	12471	12155	6893	**6672**

I-34	38405	38330	37439	**36894**	51961	51927	51682	**50904**	15810	15554	11503	**11452**

For all ACO algorithms, we set the number of ants N_{ants} = 20, the factors reflecting the relative importance of pheromone trail and heuristic information α = 1 and β = 2, the initial pheromone amount τ_0 = 0.0001, the initial heuristic value η_0 = 0.0001, the local and global pheromone evaporation parameters ρ_l = 0.40 and ρ_g = 0.35, the parameter for the pseudo-random proportional rule q_0 = 0.95, and the predefined number of iterations GEN_{max} = 20. The parameter settings above are mainly referred to [35][47]. We randomly select 12 out of 36 instances for performance comparison. The results of the average *Cost* (see Eq. (7)), the average $Power_{total}$, and the average R^B_{total} are shown in Table 3, respectively.

5.3.1 Numerical Result Analysis

A-1 is beaten by A-2 in 11 instances (except I-21) regarding the average *Cost*, in 11 instances (except I-5) regarding the average $Power_{total}$, and in 10 instances (except I-21 and I-27) regarding the average R^B_{total}, respectively. Hence, A-1 is overweighed by A-2 in terms of the overall performance, showing the effectiveness of the EBAPMS scheme. Moreover, A-2 is beaten by A-3 in all the 12 instances with respect to the average cost and the average R^B_{total}, and in all but one instance (i.e., I-9) with respect to the average $Power_{total}$, respectively. The superiority of A-3 over A-2 is because of the TVMO scheme as it is the only difference between them. Compared with A-3, A-4 wins all instances in terms of the average *Cost* and the average $Power_{total}$, and 9 instances (except I-1, I-12, and I-29) in terms of the average R^B_{total}, respectively. That shows the effectiveness of the DIEX scheme.

5.3.2 Underlying Mechanism Discussion

With two types of heuristic information developed, the EBAPMS scheme considers the power and the bandwidth resources. PMs with less power consumption are emphasized, and among them, those consuming fewer bandwidth resources are chosen with higher probabilities. That is why A-2 beats A-1 in terms of the average *Cost*, the average $Power_{total}$ and the average R^B_{total}. Besides, the comparison between A-2 and A-3 justifies the TVMO scheme. It is seen that the bandwidth resource saving is significant in most instances. That is because the TVMO scheme ensures VMs with higher traffic demand are placed earlier. Hence, these VMs are more likely to be hosted by identical or neighboring PMs, reducing network bandwidth consumption. In addition, the DIEX scheme constructs new solutions by spreading promising components of the best solution across the colony, which helps to construct high-quality solutions. Thanks to the DIEX scheme, A-4 achieves better performance than A-3. Therefore, the experimental results clearly show that the

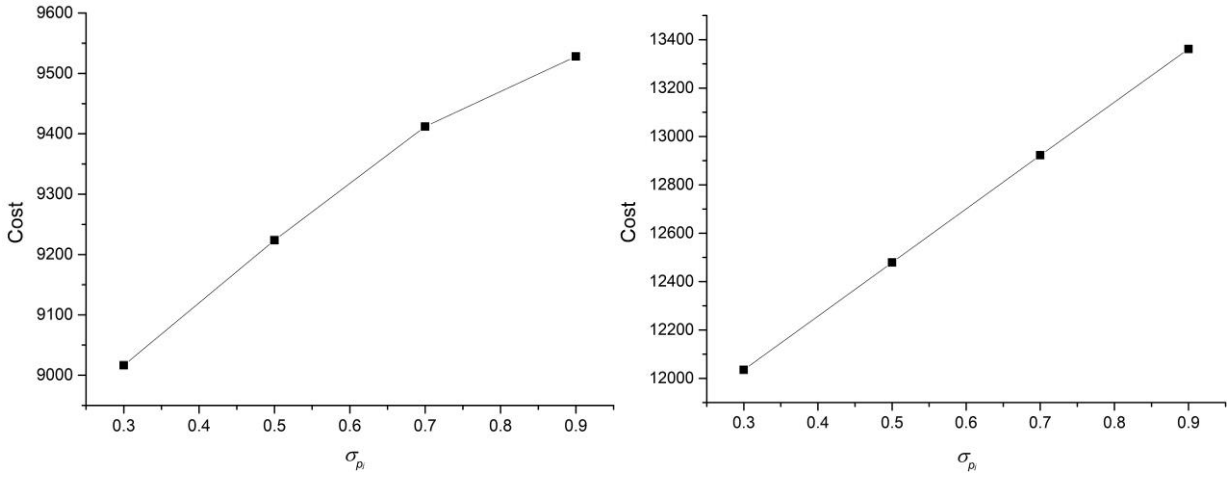
ACO's performance is incrementally improved by adding the three performance-enhancing schemes one by one, which demonstrates the effectiveness of each scheme.

5.3.3 Practical Implications of the Results

We compare the results obtained by A-1 and A-4 and analyze their practical implications. Similar conclusions can be drawn when comparing the other algorithms in Table 3. A-1 is the original ACO adapted for the VMP problem while A-4 is the ACO proposed in this paper. A-4 is an upgraded version of A-1, i.e., A-1 with the EBAPMS, TVMO, and DIEX schemes. It is easily seen that A-4 performs better than A-1 in terms of cost, power consumption and bandwidth resource consumption. For example, the power consumption of A-4's VMP solution is 812 W lower than that of A-1's in I-1. In 365 days, A-4's reduces by a power consumption of 7113.12 kW·h, compared with A-1's. Similarly, in one year, there is a gap of 9259.32 kW·h between A-1 and A-4 in I-34. Suppose 1k·W costs 0.1 $. In one year, A-4 will save as much as 711.31 $ in I-1 and 925.93 $ in I-34, respectively. A real-world datacenter may maintain tens of thousands of or even hundreds of thousands of PMs. A-4 is helpful for reducing power consumption and thus the operational cost. Besides, the bandwidth resource consumption of A-4's VMP solution is 2185 Mb/s less than that of A-1's in I-1. Using A-4 helps to reduce the congestion probability and is thus more likely to guarantee network performance. According to the Amazon AWS cloud service, we assume 2000 Mb/s in one hour costs 0.06 $. In one year, A-4 will save 525.6 $ in instance I-1, reducing the operational cost. To summarize, the proposed algorithm, i.e., A-4, performs the best in terms of power and bandwidth resource consumption as it more reasonably allocate and manage the PM and network bandwidth resources in a datacenter than A-1, A-2 and A-3.

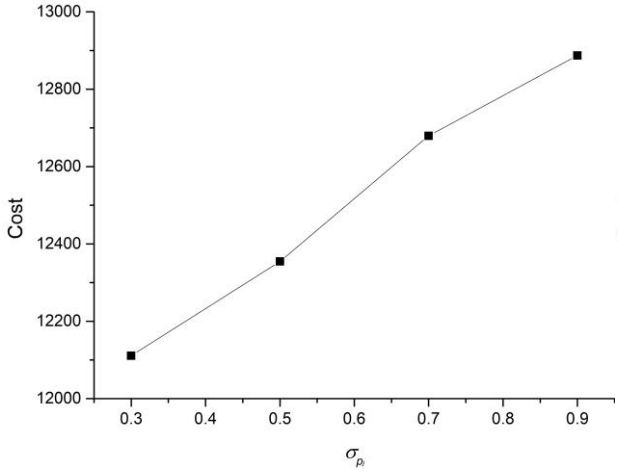
5.4 Sensitivity Analysis on the Model's Main Parameters

For the Section 2's model, the ratio of $Power(idle, p_j)$ to $Power(busy, p_j)$, σ_{p_j}, and the two weights in Eq. (7), θ_1 and θ_2 are the most important parameters. To study how sensitive these parameters are, the authors have conducted a set of experiments on σ_{p_j} and (θ_1, θ_2) and show the VMP cost values obtained. This helps reflect how the cost value defined in Eq. (7) is affected by these parameters. We randomly select 6 test instances, including I-1, I-9, I-15, I-21, I-27, and I-31.

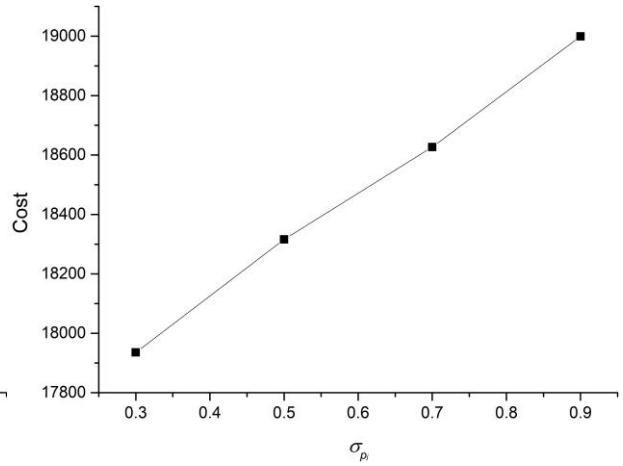


(a) I-1

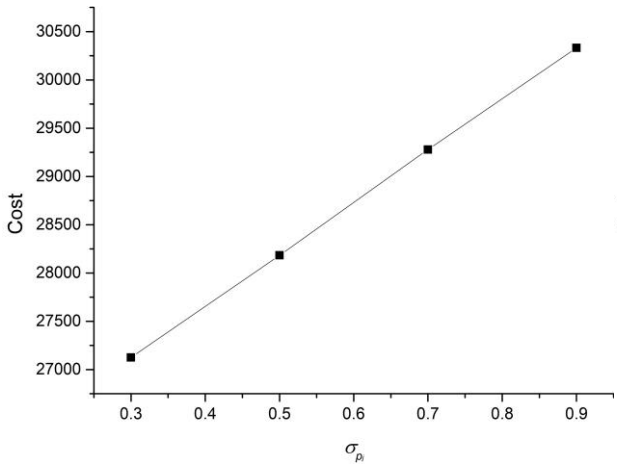
(b) I-9



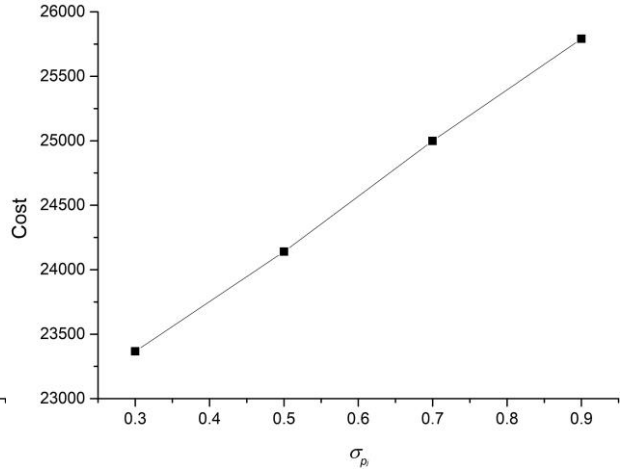
(c) I-15



(d) I-21



(c) I-27



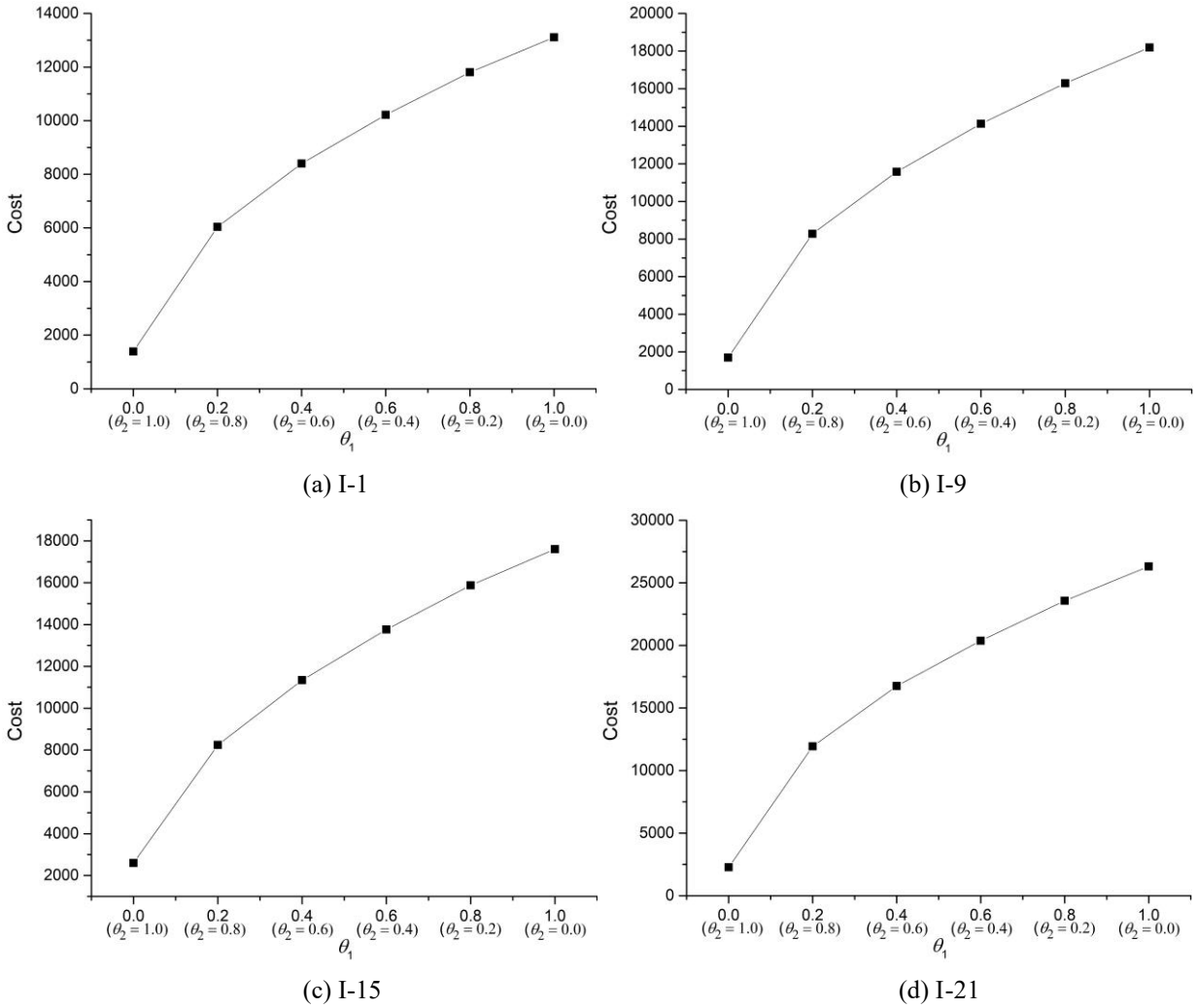
(d) I-31

Fig. 7 Results of *Cost* vs. σ_{p_j}

First of all, we study the impact of σ_{p_j} on the cost value, where we fix $\theta_1 = \theta_2 = 0.5$ as in Section 2. The value of σ_{p_j} is set to 0.3, 0.5, 0.7 and 0.9, respectively. Fig. 7 plots the results of *Cost* vs σ_{p_j}. As σ_{p_j} increases, the *Cost* value grows up linearly. Take I-1 as an example. The *Cost* value is increased by about 20% when σ_{p_j} is from 0.3 to 0.5.

Similar findings can be observed in the other five subgraphs in Fig. 7. So, σ_{p_j} influences on the *Cost* result, i.e., a larger σ_{p_j} results in a larger *Cost* value.

Then, we study the impact of θ_1 and θ_2 on the cost value, where we fix $\sigma_{p_j} = 0.7$ as in Section 2. We have $\theta_1 + \theta_2 = 1$ as specified by Constraint (8). The value of θ_1 is from 0.0 to 1.0, with an interval of 0.2. It is seen that with θ_1 increasing (while θ_2 decreasing), the *Cost* value gradually increases in Fig. 8. A smaller θ_1 causes a more significant rise in *Cost* when θ_1 is increased by 0.2, indicating that the model is more sensitive to a smaller θ_1. For example, the *Cost* value increases by 39% when θ_1 is from 0.2 to 0.4 while the cost increases by 15% when θ_1 is from 0.6 to 0.8 in Instance I-1. Similar findings can be observed in the other five subgraphs in Fig. 8. Hence, θ_1 has an impact on the *Cost* result, i.e., a larger θ_1 results in a larger *Cost* value and the model is more sensitive to a smaller θ_1.

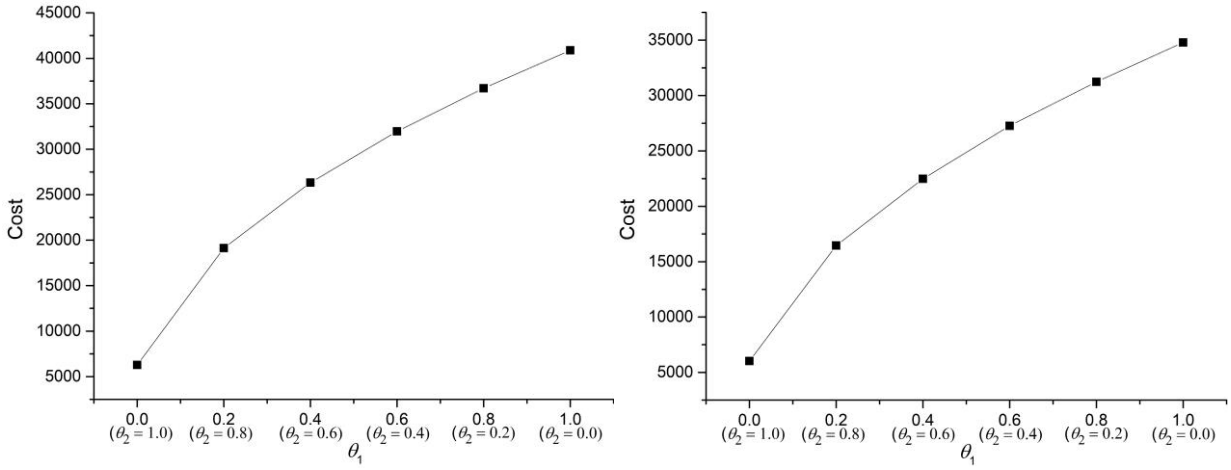


(a) I-1

(b) I-9

(c) I-15

(d) I-21



(c) I-27 (d) I-31

Fig. 8 Results of *Cost* vs. θ_1

5.5 Comparison with State-of-the-art Heuristic Algorithms

A number of heuristics have been developed for VMP problems. By utilizing problem-specific knowledge, these algorithms usually achieve decent optimization performance. This subsection compares the proposed ETA-ACO against six widely used heuristics listed below.

- MBF: the modified best fit algorithm, where PMs are sorted in descending order by their current power consumption [48]. Each VM is placed into the first PM that has sufficient resources for hosting it.
- PEBFD: the best fit decreasing algorithm. Unassigned VMs are sorted in descending order by the execution time, and they are assigned to those active PMs with the lowest energy efficiency [49].
- FF: the first fit algorithm, where each VM is deployed on the first PM with enough resources [50].
- FFD: the first fit decreasing algorithm, where PMs are sorted in decreasing order based on their bandwidth resources [29]. It helps to accommodate more VMs in one PM with enough resources.
- MBFD: the modified best fit decreasing algorithm, where a bio-inspired heuristic-based VM consolidation mechanism is developed for energy saving and QoS guarantee [51].
- PABFD: the power-aware best fit decreasing algorithm, where a target PM is selected based on the least increased power consumption and its utilization stability [24].
- ETA-ACO: the proposed ACO in this paper.

5.5.1 Numerical Result Analysis

The results of the average *Cost* are shown in Table 4. It is clearly seen that ETA-ACO outperforms all the six

heuristics since it results in the smallest average *Cost* value in each instance. ETA-ACO's average *Cost* values are at least 3% smaller. In particular, ETA-ACO obtains significantly smaller average *Cost* values (at least 8% smaller) in instances I-1, I-5, I-6, I-11, I-13, I-14, I-19, I-32, and I-33, compared with the rest of the algorithms. For example, the average *Cost* values of ETA-ACO and PEBFD are 12994 and 15518 in I-14, respectively. So, the former is 16% smaller than the latter. Meanwhile, PABFD is the second-best algorithm because it beats the other five heuristics in 17 instances, including I-2, I-4, I-6, I-8, I-10, I-12, I-13, I-14, I-15, I-20, I-22, I-24, I-28, I-29, I-30, I-31, and I-32. Besides, FFD is the worst algorithm as it leads to the largest average *Cost* values in most instances except I-7, I-8, I-9, I-35, and I-36.

Table 4 Results of the average *Cost* (Best results are in bold)

Algorithms	I-1	I-2	I-3	I-4	I-5	I-6	I-7	I-8	I-9	I-10	I-11	I-12	I-13	I-14	I-15	I-16	I-17	I-18
MBF [48]	10295	10288	10082	11047	11233	11256	13930	14110	13603	12582	12242	12605	15167	15437	13906	20419	19809	19537
PEBFD [49]	10393	10356	10025	11411	11580	11557	13957	14065	14084	12895	12215	12828	15290	14620	14087	20630	20006	19884
FF [50]	10482	10457	9978	11191	10998	11544	14552	14327	14148	12633	12275	12802	15134	14836	14172	20598	19662	19382
FFD [29]	10489	10482	10563	11870	11892	11786	14296	14244	14160	13343	13186	13317	16068	15518	14830	21148	20055	19721
MBFD [51]	10324	10339	10120	11248	11135	11416	14388	14174	14199	12675	12299	12688	14966	14565	14040	20303	19468	19400
PABFD [24]	10486	9958	10080	10814	11232	11155	14522	14043	14079	12179	12680	11910	14707	14184	13435	20596	19783	19434
ETA-ACO	**9291**	**9441**	**9249**	**10307**	**10105**	**10117**	**12977**	**12873**	**12826**	**11402**	**11100**	**11458**	**13313**	**12994**	**12755**	**19133**	**17964**	**18339**
Algorithms	I-19	I-20	I-21	I-22	I-23	I-24	I-25	I-26	I-27	I-28	I-29	I-30	I-31	I-32	I-33	I-34	I-35	I-36
MBF [48]	20473	20922	20706	24926	24069	24875	31231	31431	31528	23067	23535	23101	28480	27695	28083	38071	38725	38720
PEBFD [49]	20408	20625	20017	23651	23843	24472	31637	32036	31442	23390	23392	22638	27163	26745	26925	38624	39415	39541
FF [50]	20357	20563	19504	23608	23573	24719	31447	31797	31179	23189	23360	22402	27674	26669	26823	38326	39270	39199
FFD [29]	21660	21826	21532	25286	25512	25922	31821	31930	32038	24168	24379	23798	29202	29050	29146	38879	39329	39341
MBFD [51]	20300	20468	19725	23511	23462	24807	31589	31899	31277	23385	23155	22352	27542	26733	26973	38398	39069	39072
PABFD [24]	20946	20426	20776	23204	23756	23510	31705	31502	31426	22520	22818	22058	26501	26616	26954	38213	39268	39294
ETA-ACO	**18555**	**19047**	**18049**	**21830**	**21841**	**22650**	**29499**	**29585**	**29209**	**21068**	**21252**	**20653**	**24607**	**24435**	**24589**	**36894**	**37411**	**36971**

Table 5 Results of the average $Power_{total}$ (W) (Best results are in bold)

Algorithms	I-1	I-2	I-3	I-4	I-5	I-6	I-7	I-8	I-9	I-10	I-11	I-12	I-13	I-14	I-15	I-16	I-17	I-18
MBF [48]	14176	14192	13993	15460	15610	15459	18715	18656	18667	17194	16877	17362	19974	20737	19112	27216	26126	26519
PEBFD [49]	14350	14364	13926	15386	15556	15694	18869	18771	18763	16779	16489	17222	19488	19867	19193	26989	25736	26271
FF [50]	14426	14437	13982	15538	15266	15847	19025	18901	18940	16912	16633	17389	19662	20073	19375	26936	25658	26185
FFD [29]	14458	14472	14495	16258	16293	15966	18910	18852	18857	17588	17551	17807	20762	20863	20273	27320	26245	26607
MBFD [51]	14252	14267	14193	15584	15455	15594	18761	18642	19049	16960	16684	17128	19373	19782	19117	26587	25610	26145
PABFD [24]	13930	13788	14047	14975	15536	15037	18929	18247	18896	16087	16858	16106	19086	18846	18237	26963	25705	26002
ETA-ACO	**12995**	**13265**	**13066**	**14509**	**14268**	**14279**	**18197**	**17917**	**17957**	**15843**	**15429**	**15936**	**18631**	**18222**	**17885**	**26420**	**24876**	**25403**
Algorithms	I-19	I-20	I-21	I-22	I-23	I-24	I-25	I-26	I-27	I-28	I-29	I-30	I-31	I-32	I-33	I-34	I-35	I-36
MBF [48]	28305	28945	28807	34256	33317	33782	42165	42688	42618	31803	32454	31882	38622	38092	38347	51667	52552	52321
PEBFD [49]	28187	28543	27739	32641	32448	33378	42227	42727	42096	32282	32050	31038	36988	36451	36475	52330	53137	52834
FF [50]	28108	28478	27305	32563	32361	33305	41920	42704	41787	31819	31955	30905	36913	36116	36404	51973	52835	52621
FFD [29]	29628	30004	29930	34780	34832	34815	42670	43201	43128	33116	33247	32687	39368	39333	39583	52518	53009	52781
MBFD [51]	27960	28337	27528	32418	32212	33441	41991	42847	41869	32063	31813	30781	36752	36240	36508	52053	52548	52336
PABFD [24]	28643	27972	28633	31984	32577	31613	42351	42379	42177	30853	31231	30427	35614	36135	36574	51851	52967	52655

ETA-ACO	**26159**	**26749**	**25435**	**30475**	**30673**	**31443**	**40817**	**40995**	**40548**	**29487**	**29379**	**28939**	**34154**	**34151**	**34346**	**50904**	**51732**	**51090**

To unveil more details about the solution quality, we show the average $Power_{total}$ and average R^B_{total} results in Tables 5-6. In Table 5, ETA-ACO outperforms all the heuristics in all instances in terms of both $Power_{total}$ and R^B_{total}. In particular, ETA-ACO obtains a significantly better average R^B_{total} value in each instance. For example, in I-6, ETA-ACO gains an average R^B_{total} value of 914, 75% smaller than the second lowest value, i.e., 3799. PABFD performs better than MBF, PEBFD, FF, FFD and MBFD in 19 instances regarding $Power_{total}$, including I-1, I-2, I-4, I-6, I-8, I-10, I-12, I-13, I-14, I-15, I-18, I-20, I-22, I-24, I-26, I-28, I-29, I-30, and I-31. However, PABFD does not achieve decent performance in the average R^B_{total}, e.g., it even obtains the worst average values in Instances I-1, I-7, I-8, I-14, and I-21. On the other hand, FFD is the worst algorithm among the seven. It leads to the worst average $Power_{total}$ in 31 instances (except I-7, I-8, I-9, I-35, and I-36) and the worst average R^B_{total} in 19 instances, respectively.

Table 6 Results of the average R^B_{total} (Mb/s) (Best results are in bold)

Algorithms	I-1	I-2	I-3	I-4	I-5	I-6	I-7	I-8	I-9	I-10	I-11	I-12	I-13	I-14	I-15	I-16	I-17	I-18
MBF [48]	3316	3207	2738	2254	2951	3799	6151	7082	4649	4579	3863	4041	7819	6827	4638	9650	10109	7753
PEBFD [49]	3179	2860	2661	4867	5119	4562	5793	6578	6684	7143	5152	5701	9371	5724	5339	11082	11752	10029
FF [50]	3410	3207	1903	3007	2978	3925	7848	7300	6451	5760	4967	5042	8456	6106	5131	11089	10718	8104
FFD [29]	3316	3207	3610	4182	4166	4785	7151	7098	6740	6838	6301	6133	9236	6807	5375	12171	10752	8361
MBFD [51]	3173	3201	1839	3190	3019	4179	7876	7368	6353	5803	4916	5346	8524	5738	5364	10841	10106	8315
PABFD [24]	5086	2863	2425	3106	3307	4771	7965	7840	6276	6154	6114	4930	8265	6869	5332	11016	11045	8903
ETA-ACO	**1939**	**1521**	**607**	**1394**	**808**	**914**	**2382**	**3224**	**2560**	**2998**	**2894**	**2935**	**2708**	**2379**	**2349**	**5840**	**5156**	**5225**
Algorithms	I-19	I-20	I-21	I-22	I-23	I-24	I-25	I-26	I-27	I-28	I-29	I-30	I-31	I-32	I-33	I-34	I-35	I-36
MBF [48]	6092	6133	5258	8313	6974	9814	13147	12393	13105	7260	7383	7130	11428	9114	10333	15145	15413	16153
PEBFD [49]	6203	6008	5650	7298	9172	9148	14787	15068	14323	7213	8198	7851	10371	10094	10933	15659	16840	18316
FF [50]	6228	5888	3906	7373	8007	10623	14850	14086	14075	7936	8382	6969	13003	10868	10665	15381	17109	17442
FFD [29]	7777	7246	5607	8314	9403	11479	14299	13143	13885	8459	9127	8018	12479	11861	11496	16280	16839	17594
MBFD [51]	6514	5904	4513	7389	7955	10606	15249	14113	14267	8107	7764	7192	12899	10768	11056	15470	17075	17724
PABFD [24]	7553	7210	6592	7342	8214	10299	14724	13738	14011	7897	8122	6877	11672	10541	10742	15228	16686	17763
ETA-ACO	**2071**	**3177**	**2139**	**4936**	**3642**	**6111**	**8621**	**8362**	**7883**	**4274**	**6340**	**3958**	**6672**	**5275**	**5444**	**11452**	**11089**	**11116**

5.5.2 Underlying Mechanism Discussion

Firstly, we analyze why PABFD results in small average $Power_{total}$ but large average R^B_{total} in most instances. PABFD selects a PM based on the increased power consumption and its utilization stability, which reduces $Power_{total}$ and balances network loads. However, PABFD pays insufficient attention to network bandwidth consumption, which may result in large R^B_{total} in some instances. Secondly, we analyze why FFD does not perform well in both $Power_{total}$ and R^B_{total}. As we know, FFD is a first fit decreasing algorithm that sorts PMs based on their bandwidth resources. This algorithm

aims to host as many VMs on one PM as possible. However, it does not carefully consider power consumption. Also, FFD ignores the communication demand between VMs. So, it is not good at lower the network bandwidth consumption.

With problem-specific knowledge, heuristics usually provide sufficiently good solutions within a limited time. These algorithms can respond quickly to the VMP task. They are, however, limited by the ways they explore the search space. They aim to find an acceptable solution as quickly as possible, rather than improve solution quality towards the optima. It makes sense that the six heuristics do not achieve promising optimization results. On the other hand, as a widely used metaheuristic, ACO is good at global exploration and local exploitation. ETA-ACO is a modified version of ACO for VMP. By incorporating specially devised problem-specific schemes, ETA-ACO achieves smaller $Power_{total}$ and R^{B}_{total} values than those heuristics.

5.6 Comparison with State-of-the-art Metaheuristics

This subsection compares ETA-ACO with the following seven state-of-the-art metaheuristics.

- ACS: the ant colony system algorithm that tackles the energy-efficient VMP problem, where problem-specific heuristics are devised based on PM usage and capacity [18].
- PPVMP: the power-aware and performance-guaranteed VMP algorithm based on ACO [19]. The heuristic information measures how power and performance degradation is alleviated.
- OEMACS: the order exchange and migration ant colony system for homogeneous and heterogeneous server environments, where local search techniques are integrated to improve solution quality [35].
- GATA: the genetic algorithm with tabu search for the VMP problem, where energy efficiency and load balancing are critical factors for consideration [52].
- IGGA: the improved grouping genetic algorithm that reduces migration costs and power consumption simultaneously. A greedy heuristic and a swap operation are used to increase the consolidation score [53].
- MSBPP: the modified squirrel search algorithm for addressing the one-dimensional bin packing problem. MSBPP is based on an improved best fit heuristic that generates eligible initial packing of items [54]. That we select MSBPP for performance comparison is because VMP also belongs to the bin packing problem.
- PSO: the improved particle swarm optimization for the energy- and QoS-aware VMP problem [55]. PSO adopts a local fitness-first strategy to update particle positions.
- ETA-ACO: the proposed ACO in this paper.

ACS, PPVMP, OEMACS and ETA-ACO are ACO-based algorithms. For a fair completion, they use the same

parameter settings in Subsection 5.3. For GATA, IGGA, MSBPP and PSO, their parameter settings in [52][53][54][55] are adopted. To be specific, we set N_{pop} = 20, $GEN_{\max}$ = 50, *tsGen* = 15 and *neighborLen* = 20 for GATA; we set the population size N_{pop} = 75, the maximum generation number $GEN_{\max}$ = 100, the crossover probability $prob_c$ = 0.5, and the mutation probability $prob_m$ = 0.1 for IGGA; we set N_{pop} = 50, *Pdp* = 0.1 and $GEN_{\max}$ = 100 for MSBPP; we set N_{pop} = 20 and $GEN_{\max}$ = 30 for PSO.

5.6.1 Numerical Result Analysis

The results of the average *Cost* are shown in Table 7. ETA-ACO performs the best among the eight EAs since it obtains the smallest average *Cost* in 36 instances. GATA and PSO achieve slightly better performance than ACS, PPVMP, OEMACS, IGGA, and MSBPP in a number of instances, such as I-4, I-9, I-15, I-16, I-20, I-23, I-24, I-25, I-26, and I-36. PPVMP seems the one with the worst performance since it obtains the largest and second-largest average *Cost* values in 12 and 15 instances, respectively.

Table 7 Mean values of *Cost* (Best results are in bold)

Algorithms	I-1	I-2	I-3	I-4	I-5	I-6	I-7	I-8	I-9	I-10	I-11	I-12	I-13	I-14	I-15	I-16	I-17	I-18
ACS [18]	10226	10124	10077	10855	10847	11395	14235	14209	13592	12506	12850	11864	14795	13782	14341	21056	19711	19640
PPVMP [19]	10217	9890	10044	11050	11161	11509	14494	14276	13865	13263	13360	13207	15023	14714	14543	20505	21762	19518
OEMACS [35]	9787	10028	9883	10977	10454	10865	14826	15596	14871	11943	11697	11878	14887	14295	14090	20416	19780	18598
GATA [52]	10030	10033	9646	10822	10674	11040	13495	13543	13418	12037	11656	12221	14229	14110	13623	20143	18995	19026
IGGA [53]	10063	10074	9626	10952	10687	11010	13328	13193	13122	12399	11913	12514	14709	14417	13991	20226	19722	19667
MSBPP [54]	10179	10112	9710	11064	10842	11256	13995	14286	13865	12599	11885	12520	14446	14735	14107	20918	20304	20077
PSO [55]	10225	10203	9816	10833	10791	11421	13657	13644	13175	12390	11734	12529	14584	13983	13821	20188	19071	19173
ETA-ACO	**9291**	**9441**	**9249**	**10307**	**10105**	**10117**	**12977**	**12873**	**12826**	**11402**	**11100**	**11458**	**13313**	**12994**	**12755**	**19133**	**17964**	**18339**
Algorithms	I-19	I-20	I-21	I-22	I-23	I-24	I-25	I-26	I-27	I-28	I-29	I-30	I-31	I-32	I-33	I-34	I-35	I-36
ACS [18]	19683	21796	19764	23170	23198	25456	32240	31887	31977	23384	23340	22886	26402	27426	26834	39587	38915	38930
PPVMP [19]	20448	21660	20963	23999	24099	24637	33814	31158	29994	23149	23809	23053	27564	26589	26999	38145	39039	38948
OEMACS [35]	21743	21423	20642	25573	26195	27078	34733	33160	33432	22006	22304	21845	25268	25420	25849	40776	40383	39078
GATA [52]	19781	20015	19223	23075	23053	23840	30669	31082	30542	22630	22731	21912	26754	25968	26027	37960	38586	38438
IGGA [53]	20038	20274	19402	23310	23580	24510	32277	31690	31239	22982	23237	22249	27000	26208	26294	37996	38661	38509
MSBPP [54]	20119	20314	19455	23583	23150	24617	30907	31331	30846	23204	23312	22377	27913	26933	26902	38075	38815	38710
PSO [55]	19841	20170	18613	23196	22604	23459	30765	30555	30751	22424	22811	21806	26425	25665	25919	38094	38711	38413
ETA-ACO	**18555**	**19047**	**18049**	**21830**	**21841**	**22650**	**29499**	**29585**	**29209**	**21068**	**21252**	**20653**	**24607**	**24435**	**24589**	**36894**	**37411**	**36971**

To discover more details about the solution quality, we show the corresponding average $Power_{total}$ and average R^B_{total} results in Tables 8-9. Clearly, ETA-ACO outperforms the other EAs in 33 instances (except I-18, I-27, and I-31) regarding $Power_{total}$ and in all 36 instances regarding R^B_{total}, respectively. The significance of results shows that ETA-ACO is good at reducing both $Power_{total}$ and R^B_{total}. Especially, in some instances, the average R^B_{total} obtained by

ETA-ACO is significantly smaller than those obtained by the others. For example, in instances I-3, I-5, I-6, I-7, I-13, I-14, and I-19, ETA-ACO leads to at least 50% less bandwidth resource consumption. Therefore, the proposed algorithm gains excellent optimization performance in terms of power and bandwidth resource consumption. In Table 8, PPVMP obtains worse results than the others in 15 instances, e.g., I-3, I-10, I-13, I-15, I-17, I-28, I-30, I-33, and I-36. In Table 9, MSBPP is the worst EA in half of the instances, including I-1, I-4, I-14, I-15, I-16, I-18, I-19, I-21, I-22, I-24, I-28, I-29, I-30, I-31, I-32, I-33, I-34, and I-35.

Table 8 Mean values of $Power_{total}$ (W) (Best results are in bold)

Algorithms	I-1	I-2	I-3	I-4	I-5	I-6	I-7	I-8	I-9	I-10	I-11	I-12	I-13	I-14	I-15	I-16	I-17	I-18
ACS[18]	14280	14005	13996	14865	14888	14944	19040	19471	18737	16842	17424	16385	19909	18945	19735	28220	26196	26574
PPVMP [19]	14230	13837	14030	15282	15355	15700	19088	18625	18488	17365	17616	17548	20820	20226	19807	26780	25840	25764
OEMACS [35]	13647	14020	13898	15329	14685	15193	19280	19795	19302	16475	16091	16398	19259	19183	19058	26820	25764	**25211**
GATA [52]	14018	14053	13576	15046	14906	15391	18398	18400	18430	16452	16152	16882	18944	19262	18711	26628	25081	25648
IGGA [53]	13987	14020	13493	15068	14797	15124	18328	17929	18082	16534	16188	16899	19001	19061	18727	26679	25757	25781
MSBPP [54]	14034	14042	13587	15110	14960	15327	18447	18416	18428	16454	16121	16867	18969	19343	18781	26659	25070	25609
PSO [55]	14109	14152	13691	15201	15145	15516	18633	18599	17990	16411	16320	16988	18802	18604	19027	26669	25255	25848
ETA-ACO	**12995**	**13265**	**13066**	**14509**	**14268**	**14279**	**18197**	**17917**	**17957**	**15843**	**15429**	**15936**	**18631**	**18222**	**17885**	**26420**	**24876**	25403
Algorithms	I-19	I-20	I-21	I-22	I-23	I-24	I-25	I-26	I-27	I-28	I-29	I-30	I-31	I-32	I-33	I-34	I-35	I-36
ACS [18]	27101	30289	27696	32240	32122	35000	43041	43353	43452	32169	31888	31592	35470	37251	36389	53674	52735	52201
PPVMP [19]	28570	29290	29141	33168	33119	33472	41895	41705	**40401**	32119	32664	31843	37703	36544	37164	51552	52586	52527
OEMACS [35]	30154	30003	28675	35547	36065	37536	47351	44755	45113	30296	30219	29923	**33956**	34616	35155	55647	54557	52198
GATA [52]	27584	27951	26896	32050	31881	32750	41245	42105	41259	31333	31449	30434	36426	35585	35640	51402	52182	51649
IGGA [53]	27541	27963	26855	31788	31855	32805	42105	42426	41661	31161	31416	30363	36198	35607	35664	51550	52023	51334
MSBPP [54]	27601	27939	26896	32056	31870	32784	41255	42128	41267	31387	31460	30424	36404	35603	35725	51377	52199	51733
PSO [55]	27732	28153	26080	32267	31304	32332	41338	41244	41532	31304	31413	30546	36036	35126	35482	51583	52408	51580
ETA-ACO	**26159**	**26749**	**25435**	**30475**	**30673**	**31443**	**40817**	**40995**	40548	**29487**	**29379**	**28939**	34154	**34151**	**34346**	**50904**	**51732**	**51090**

Table 9 Mean values of R_{total}^{B} (Mb/s) (Best results are in bold)

Algorithms	I-1	I-2	I-3	I-4	I-5	I-6	I-7	I-8	I-9	I-10	I-11	I-12	I-13	I-14	I-15	I-16	I-17	I-18
ACS [18]	2233	2895	2552	3708	3516	5973	6446	4953	4277	5358	5101	3599	6404	4527	4656	9459	9456	8065
PPVMP [19]	2385	1937	2027	3143	3600	4232	7381	7730	6464	7003	6737	6283	4185	4849	5517	11085	16648	9898
OEMACS [35]	2232	2053	1354	2313	1627	2134	8116	9649	8292	3582	3717	3526	8323	6267	5688	10673	10883	7479
GATA [52]	2133	1929	1304	2779	2360	2585	5048	5298	4513	4353	3263	3687	6764	5187	4575	10114	9613	8125
IGGA [53]	2591	2503	1779	3565	3058	3666	4355	5093	4153	5805	4652	5238	8445	7222	6378	10240	10575	10392
MSBPP [54]	3176	2690	1968	4033	3338	4267	7157	8293	6686	6820	4730	5364	7527	7728	6715	12813	13995	12256
PSO [55]	3137	2786	2249	1880	1832	4450	5068	5121	4446	6093	2976	5008	8417	6691	4460	10187	9457	8184
ETA-ACO	**1939**	**1521**	**607**	**1394**	**808**	**914**	**2382**	**3224**	**2560**	**2998**	**2894**	**2935**	**2708**	**2379**	**2349**	**5840**	**5156**	**5225**
Algorithms	I-19	I-20	I-21	I-22	I-23	I-24	I-25	I-26	I-27	I-28	I-29	I-30	I-31	I-32	I-33	I-34	I-35	I-36
ACS [18]	6312	5701	3745	5844	6652	8399	15022	12401	12514	7634	8504	7015	11647	10790	10752	15905	15734	17492
PPVMP [19]	4464	8847	5438	7182	8025	9629	22996	14205	12910	6321	8157	6970	9892	8842	8748	15872	16810	16575
OEMACS [35]	5975	4195	5466	6632	8439	7564	13045	13983	14112	7061	8854	7532	11113	9681	9987	15115	16874	18148
GATA [52]	4646	4451	3944	6134	6812	8003	13411	12616	12776	6509	6650	5820	10221	9066	9192	15481	15962	16948
IGGA [53]	6665	6326	5614	8728	9850	11182	17497	14432	14627	9220	9635	8244	12147	10276	10508	15161	16817	18184

MSBPP [54]	6894	6676	5780	9196	7456	11701	14429	13721	14131	9559	9843	8701	15256	13524	13072	16111	16976	17901
PSO [55]	4228	4548	3537	5892	6453	7409	13560	12881	12886	5051	7334	4220	9870	9126	9177	15541	15822	17044
ETA-ACO	**2071**	**3177**	**2139**	**4936**	**3642**	**6111**	**8621**	**8362**	**7883**	**4274**	**6340**	**3958**	**6672**	**5275**	**5444**	**11452**	**11089**	**11116**

To further support the above observations, we use the Friedman test for performance comparison with respect to *Cost*. The rankings of the eight EAs are collected in Table 12, with all 36 test instances considered. ETA-ACO, GATA, and PSO take the first, second, and third places in the competition, respectively, while PPVMP and ACS are two worst. The reason behind this is discussed in Subsection 5.4.2.

Table 12 Rankings of the eight EAs

Algorithm	ACS [18]	PPVMP [19]	OEMACS [35]	GATA [52]	IGGA [53]	MSBPP [54]	PSO [55]	ETA-ACO
Average rank	5.89	6.72	5.25	3.00	4.53	5.81	3.81	1.00
Position	7	8	5	2	4	6	3	1

Computational time is a direct indicator reflecting the time complexity of an algorithm. Table 13 shows the average computational time (ACT) of each algorithm. ACT values gradually increase with the increasing numbers of VMs and PMs. ACS and PSO are the two most time-consuming algorithms in large instances, while IGGA is the one with the least ACT values in most instances. ETA-ACO is somewhere in the middle. Considering its promising performance in all test instances, one can see ETA-ACO achieves a decent overall optimization performance.

Table 13 Results of ACT (*sec*.)

Algorithms	I-1	I-2	I-3	I-4	I-5	I-6	I-7	I-8	I-9	I-10	I-11	I-12	I-13	I-14	I-15	I-16	I-17	I-18
ACS [18]	0.23	0.20	0.20	0.42	0.44	0.39	1.15	1.10	1.16	1.59	1.60	1.58	3.18	3.64	3.28	9.41	9.65	9.36
PPVMP [19]	0.05	0.05	0.05	0.08	0.09	0.08	0.26	0.26	0.27	0.32	0.33	0.32	0.66	0.66	0.66	2.21	2.25	2.24
OEMACS [35]	0.08	0.08	0.08	0.28	0.30	0.36	1.08	0.53	0.54	0.58	0.57	0.57	1.24	1.26	1.27	4.27	4.33	4.30
GATA [52]	0.30	0.29	0.28	0.59	0.59	0.59	2.20	2.20	2.20	1.00	1.01	1.01	2.21	2.28	2.26	10.72	9.23	9.16
IGGA [53]	0.12	0.14	0.16	0.17	0.13	0.14	0.45	0.45	0.45	0.23	0.23	0.23	0.46	0.46	0.46	1.93	1.75	1.71
MSBPP [54]	0.24	0.08	0.16	0.24	0.16	0.24	0.80	0.80	0.80	0.32	0.40	0.32	0.88	0.72	0.80	3.68	4.40	4.00
PSO [55]	0.07	0.06	0.07	0.15	0.13	0.14	2.95	3.03	3.06	0.31	0.27	0.33	0.84	0.88	0.92	15.89	14.26	15.42
ETA-ACO	0.15	0.16	0.13	0.26	0.26	0.25	0.59	0.66	0.64	1.01	0.93	0.98	1.85	1.79	1.81	4.53	4.81	4.44
Algorithms	I-19	I-20	I-21	I-22	I-23	I-24	I-25	I-26	I-27	I-28	I-29	I-30	I-31	I-32	I-33	I-34	I-35	I-36
ACS [18]	5.67	7.31	9.07	19.31	15.76	14.47	36.63	31.67	32.61	12.46	13.24	13.31	25.40	26.25	25.99	75.68	75.79	85.99
PPVMP [19]	1.09	1.11	1.11	2.33	2.34	2.29	8.21	9.62	9.92	3.19	3.20	3.22	6.87	6.91	6.94	24.34	24.25	25.14
OEMACS [35]	1.86	1.88	1.91	3.87	3.96	3.79	12.27	12.12	12.88	4.71	4.71	4.72	10.07	10.12	10.08	32.67	32.90	34.35
GATA [52]	2.45	2.31	2.33	5.29	5.38	5.58	22.10	21.06	21.62	4.78	6.33	4.51	9.79	9.89	9.81	38.67	40.06	41.53
IGGA [53]	0.48	0.48	0.49	1.02	1.03	1.02	3.93	3.95	3.95	0.84	0.83	0.83	1.80	1.80	1.82	6.95	6.98	6.98
MSBPP [54]	0.96	0.96	1.20	2.16	2.08	2.24	9.12	11.52	11.12	2.24	2.08	1.52	3.28	3.04	4.64	24.80	17.68	13.92
PSO [55]	3.65	3.63	3.32	9.04	7.24	5.61	14.61	16.40	19.76	7.06	5.54	7.22	10.19	9.90	10.39	93.36	93.00	91.04
ETA-ACO	3.22	3.31	3.20	6.55	6.12	6.57	17.48	17.18	16.60	7.79	8.08	8.14	15.81	15.20	15.94	35.94	34.38	34.34

5.6.2 Underlying Mechanism Discussion

Firstly, we analyze why ETA-ACO, GATA, and PSO obtain decent optimization performance. The proposed EBAPMS and TVMO schemes consider power and bandwidth factors simultaneously. That helps ETA-ACO to reduce the power and bandwidth resource consumption effectively. The DIEX scheme spreads the promising features found over the ant colony by allowing partial solution exchange. This scheme complements the pheromone-based indirect information exchange among ants and thus helps ETA-ACO to construct high-quality solutions. As for GATA, incorporating tabu search into the genetic algorithm is beneficial to energy efficiency and load balancing. However, maintaining and looking up a tabu list incurs additional overhead, especially when the problem scale is relatively large. PSO adopts a two-dimensional structure for particle encoding and a fitness-first heuristic for the particle position update. Besides, its parameters and operators are modified to adapt to the energy- and QoS-aware VMP problem. Hence, PSO is also good at tackling the VMP problem concerned in this paper. Nevertheless, PSO still suffers from prematurity and high computational cost due to its intrinsic mechanism and local search operator.

Secondly, we analyze why PPVMP and ACS are losers in the competition. PPVMP aims to reduce power consumption while ensuring VM performance, seeking a balance between them. Subject to the VM performance guarantee, PPVMP usually leads to higher power consumption than those whose objective is to minimize power consumption. As for ACS, integrating problem-specific heuristics helps to improve PM usage and capacity. However, these heuristics are computationally expensive. That is why ACS is usually among those with the larger ACT values. In addition, neither PPVMP nor ACS pays enough attention to reducing bandwidth resource consumption. The above explains the reasons that PPVMP and ACS are the worst-performance competitors.

6. Conclusions and Future Work

This paper formulates a new variant of the virtual machine (VM) placement problem, where both the power consumption and the network resource utilization are taken into account. Heterogeneous switches and servers are considered, where each switch port can support a number of data rate modes. The total power consumption and the total bandwidth resource consumption are two objectives to minimize. To address the problem, we propose an ant colony optimization algorithm with three performance-enhancing schemes. The energy- and bandwidth-aware physical machine (PM) selection scheme considers energy-saving and bandwidth resource utilization simultaneously. The traffic-based VM

ordering scheme stores VMs in descending order according to the traffic demands among them. The direct information exchange scheme constructs high-quality solutions by information exchange among solutions. Experimental results demonstrate that the proposed ACO achieves better performance than six existing heuristics and seven metaheuristics regarding the power and bandwidth resource consumption.

The test instances concerned in this paper are quite small since a real-world data center may maintain hundreds of thousands of PMs. Due to the limited capacity of our computer, we cannot simulate large scale problems at this stage. We plan to continue our research work on VMP with larger sizes in the future. ACO is one of the most well-known evolutionary algorithms (EAs) in the literature. Our future work will also consider applying emerging EAs to the VMP problem [56]. In addition, our VMP problem aggregates two objectives, namely the power consumption and the bandwidth resource consumption, into one objective. In the future, we will consider goal programming to combine the two objectives. On the other hand, it is also interesting to study whether there is a trade-off between the two objectives. If so, we can also treat the problem above as a bi-objective optimization problem. By solving it, we can provide the decision-maker with a set of nondominated VMP solutions if he or she needs to study the relationship between the two objectives and get some intuitive view for system performance improvement. That helps to design an appropriate data center system. We plan to extend our work to a bi-objective optimization problem and develop multiobjective EAs to address it.

Acknowledgment

This work was supported in part by National Natural Science Foundation of China (No. 62002300, No. 62172342), China Postdoctoral Science Foundation (No. 2019M660245, No. 2019M663552, No. 2020T130547), and China Scholarship Council, P. R. China.

References

[1] K. Zheng, X. Wang, L. Li, and X. Wang, “Joint Power Optimization of Data Center Network and Servers With Correlation Analysis,” in *Proc. INFOCOM 2014*, Toronto, Canada, pp. 2598-2606, 2014.

[2] P. Garraghan, X. Ouyang, R. Yang, D. McKee and J. Xu, “Straggler Root-Cause and Impact Analysis for Massive-scale Virtualized Cloud Datacenters,” *IEEE Transactions on Services Computing*, vol. 12, no. 1, pp. 91-104, Jan. 2019.

[3] S. Benbrahim, A. Quintero and M. Bellaïche, “Live Placement of Interdependent Virtual Machines to Optimize Cloud Service Profits and Penalties on SLAs,” *IEEE Transactions on Cloud Computing*, vol. 7, no. 1, pp. 237-249, Jan. 2019.

[4] C. Huang, Y. Li, and X. Yao, “A Survey of Automatic Parameter Tuning Methods for Metaheuristics,” *IEEE Transactions on Evolutionary Computation*, vol. 24, no. 2, pp. 201-216, 2020.

[5] A. Rajasekhar, N. Lynn, S. Das, and P.N. Suganthan, “Computing with The Collective Intelligence of Honey Bees,” *Swarm and Evolutionary Computation*, vol. 32, pp. 25-48, 2017.

[6] G. Liu, Y. Li, L. Jiao, Y. Chen, and R. Shang, “Multiobjective Evolutionary Algorithm Assisted Stacked Autoencoder for PoISAR Image Classification,” *Swarm and Evolutionary Computation*, published online, 2020. DOI: https://doi.org/10.1016/j.swevo.2020.100794

[7] B. H. Nguyen, B. Xue, P. Andreae, H. Ishibuchi, and M. Zhang, “Multiple Reference Points-Based Decomposition for Multiobjective Feature Selection in Classification: Static and Dynamic Mechanisms,” *IEEE Transactions on Evolutionary Computation*, vol. 24, no. 1, pp. 170-184, 2020.

[8] E. H. Houssein, A. G. Gad, Y. M. Wazery, and P. N. Suganthan, “Task Scheduling in Cloud Computing based on Meta-heuristics: Review, Taxonomy, Open Challenges, and Future Trends,” *Swarm and Evolutionary Computation*, published online, 2021. DOI: https://doi.org/10.1016/j.swevo.2021.100841

[9] S. Mnasri, N. Nasri, A. Van Den Bossche, and T. Val, “A Comparative Analysis With Validation of NSGA-III and MOEA/D in Resolving the 3D Indoor Redeployment Problem in DL-IoT,” in *Proc. 2017 International Conference on Internet of Things, Embedded Systems and Communications (IINTEC 2017)*, Gafsa, Tunisia, pp. 15-20, 2017.

[10] S. Mnasri, F. Abbes, K. Zidi, and K. Ghedira, “A Multi-objective Hybrid BCRC-NSGAII Algorithm to Solve the VRPTW,” in *Proc. 13th International Conference on Hybrid Intelligent Systems (HIS 2013)*, Gammarth, Tunisia, pp. 60-65, 2013.

[11] A. S. Abohamama and E. Hamouda, “A Hybrid Energy–Aware Virtual Machine Placement Algorithm for Cloud Environments,” *Expert Systems with Applications*, vol. 150, Jul. 2020.

[12] N. K. Sharma and G. R. M. Reddy, "Multi-Objective Energy Efficient Virtual Machines Allocation at the Cloud Data Center," *IEEE Transactions on Services Computing*, vol. 12, no. 1, pp. 158-171, Jan. 2019.

[13] M. Dorigo, M. Birattari, and T. Stutzle, “Ant colony optimization,” *IEEE Computational Intelligence Magazine,* vol. 1, no. 4, pp. 28-39, Nov. 2006.

[14] M. Mavrovouniotis, F. M. Muller, and S. Yang, “Ant Colony Optimization With Local Search for Dynamic Traveling Salesman Problems,” *IEEE Transactions on Cybernetics,* vol. 47, no. 7, pp. 1743-1756, Jul. 2016.

[15] Z. Wang, H. Xing, T. Li, Y Yang, R. Qu, and Y. Pan, “A Modified Ant Colony Optimization Algorithm for Network Coding Resource Minimization,” *IEEE Transactions on Evolutionary Computation,* vol. 20, no. 3, pp. 325-342, Jun. 2016.

[16] J. Eaton, S. Yang, and M. Gongora, “Ant Colony Optimization for Simulated Dynamic Multi-Objective Railway Junction Rescheduling,” *IEEE Transactions on Intelligent Transportation Systems,* vol. 18, no. 11, pp. 2980-2992, Nov. 2017.

[17] Z. Zhao, M. Hou, N. Zhang, and M Gao, “Multipath Routing Algorithm Based on Ant Colony Optimization and Energy Awareness,” *Wireless Personal Communications,* vol. 94, no. 4, pp. 2937- 2948, Jun. 2017.

[18] F. Alharbi, Y. Tian, M. Tang, W. Zhang, C. Peng and M. Fei, “An Ant Colony System for energy-efficient dynamic Virtual Machine Placement in data centers,” *Expert Systems with Applications*, Vol. 120, pp. 228-238, Apr. 2019.

[19] H. Zhao, J. Wang, F. Liu, Q. Wang, W. Zhang and Q. Zheng, "Power-Aware and Performance-Guaranteed Virtual Machine Placement in the Cloud," *IEEE Transactions on Parallel and Distributed Systems*, vol. 29, no. 6, pp. 1385-1400, Jun. 2018.

[20] W. Fang, X. Liang, S. Li, L. Chiaraviglio, and N. Xiong, “VMPlanner: Optimizing virtual machine placement and traffic flow routing to reduce network power costs in cloud data centers,” *Computer Networks,* vol. 57, no. 1, pp. 179-196, Jan. 2013.

[21] M. Gaggero and L. Caviglione, "Model Predictive Control for Energy-Efficient, Quality-Aware, and Secure Virtual Machine Placement," *IEEE Transactions on Automation Science and Engineering*, vol. 16, no. 1, pp. 420-432, Jan. 2019.

[22] F. Farahnakian, T. Pahikkala, P. Liljeberg, J. Plosila, N. T. Hieu and H. Tenhunen, "Energy-Aware VM Consolidation in Cloud Data Centers Using Utilization Prediction Model," *IEEE Transactions on Cloud Computing*, vol. 7, no. 2, pp. 524-536, Apr. 2019.

[23] Y. Chen, X. Chen, W. Liu, Y. Zhou, A. Y. Zomaya, R. Ranjan and S. Hu, “Stochastic scheduling for variation-aware virtual machine placement in a cloud computing CPS,” *Future Generation Computer Systems*, vol. 105, pp. 779-788, Apr. 2020.

[24] N. T. Hieu, M. D. Francesco and A. Ylä-Jääski, "Virtual Machine Consolidation with Multiple Usage Prediction for Energy-Efficient Cloud Data Centers," *IEEE Transactions on Services Computing*, vol. 13, no. 1, pp. 186-199, Jan. 2020.

[25] A. R. Ilkhechi, and I. Korpeoglu, “Network-aware virtual machine placement in cloud data centers with multiple traffic-intensive components,” *Computer Networks,* vol. 91, pp. 508-527, Nov. 2015.

[26] T. Fukunaga, S. Hirahara, and H. Yoshikawa, “Virtual machine placement for minimizing connection cost in data center networks,” in *Proc. IEEE INFOCOM*, Hong Kong, China, pp. 486-491, Aug. 2015.

[27] R. Wang, J. A. Wickboldt, R. P. Esteves, L. Shi, B. Jennings, and L. Z. Granville, “Using Empirical Estimates of Effective Bandwidth in Network-Aware Placement of Virtual Machines in Datacenters,” *IEEE Transactions on Network and Service Management,* vol. 13, no. 2, pp. 267-280, Jun. 2016.

[28] Y. Guo, A. L. Stolyar and A. Walid, "Shadow-Routing Based Dynamic Algorithms for Virtual Machine Placement in a Network Cloud," *IEEE Transactions on Cloud Computing*, vol. 6, no. 1, pp. 209-220, Jan. 2018.

[29] J. Son and R. Buyya, "Priority-Aware VM Allocation and Network Bandwidth Provisioning in Software-Defined Networking (SDN)-Enabled Clouds," *IEEE Transactions on Sustainable Computing*, vol. 4, no. 1, pp. 17-28, Jan. 2019.

[30] R. A. C. D. Silva, and N. L. S. D. Fonseca, “Topology-Aware Virtual Machine Placement in Data Centers,” *Journal of Grid Computing,* vol. 14, no. 1, pp. 75-90, Mar. 2016.

[31] T. Yang, H. Pen, W. Li, and A. Zomaya, “An energy-efficient virtual machine placement and route scheduling scheme in data center networks,” *Future Generation Computer Systems,* vol. 77, pp. 1-11, Dec. 2017.

[32] A. S. Chakravarthy and C. Sudhakar, T. Ramesh, “Energy efficient VM scheduling and routing in multi-tenant cloud data center,” *Sustainable Computing: Informatics and Systems*, vol. 22, pp. 139-151, Jun. 2019.

[33] A. Gopu, and N. Venkataraman, “Optimal VM placement in distributed cloud environment using MOEA/D,” *Soft Computing*, vol. 23, pp. 11277–11296, Nov. 2019.

[34] X. Liu, Z. Zhan, K. Du, and W. Chen, “Energy aware virtual machine placement scheduling in cloud computing based on ant colony optimization approach,” in *Proc. Genetic and Evolutionary Computation Conference (GECCO)*, Vancouver, BC, Canada, pp. 41-48, Jul. 2014.

[35] X. Liu, Z. Zhan, J. D. Deng, Y. Li, T. Gu and J. Zhang, "An Energy Efficient Ant Colony System for Virtual Machine Placement in Cloud Computing," *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 1, pp. 113-128, Feb. 2018.

[36] W. Wei, H. Gu, W. Lu, T. Zhou and X. Liu, "Energy Efficient Virtual Machine Placement with an Improved Ant Colony Optimization Over Data Center Networks," *IEEE Access*, vol. 7, pp. 60617-60625, Apr. 2019.

[37] C. E. Leiserson, “Fat-trees: Universal networks for hardware-efficient supercomputing,” *IEEE Transactions on Computers,* vol. C-34, no. 10, pp. 892-901, Oct. 1985.

[38] A. Zhou, S. Wang, B. Cheng, Z. Zheng, F. Yang, R. N. Chang, M. R. Lyu, and R. Buyya, “Cloud Service Reliability Enhancement via Virtual Machine Placement Optimization,” *IEEE Transactions on Services Computing,* vol. 10, no. 6, pp. 902-913, Dec. 2017.

[39] G. Dasgupta, A. Sharma, A. Verma, A. Neogi, and R. Kothari, “Workload management for power efficiency in virtualized data centers,” *Communications of the Acm,* vol. 54, no. 7, pp. 131-141, Jul. 2011.

[40] D. Kusic, J. O. Kephart, J. E. Hanson, N. Kandasamy, and G. Jiang, “Power and performance management of virtualized computing environments via lookahead control,” *Cluster Computing,* vol. 12, no. 1, pp. 1-15, Mar. 2009.

[41] C. Gunaratne, K. Christensen, and B. Nordman, “Managing energyconsumption costs in desktop pcs and lan switches with proxying, splittcp connections, and scaling of link speed,” *International Journal of Network Management,* vol. 15, no. 5, pp. 297–310, Oct. 2005.

[42] P. Mahadevan, P. Sharma, S. Banerjee, and P. Ranganathan, “A Power Benchmarking Framework for Network Devices,” in *Proc. IFIP International Federation for Information Processing (NETWORKING 2009)*, LNCS 5550, Aachen, Germany, pp.795-808, May. 2009.

[43] L. Yu, H. Shen, Z. Cai, L. Liu, and C. Pu, “Towards Bandwidth Guarantee for Virtual Clusters Under Demand Uncertainty in Multi-Tenant Clouds,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 29, no. 2, pp. 450-465, Feb. 2018.

[44] X. Meng, V. Pappas, and L. Zhang, “Improving the Scalability of Data Center Networks with Traffic-aware Virtual Machine Placement,” in *Proc. IEEE INFOCOM,* pp. 1-9, San Diego, CA, USA , May. 2010.

[45] R. E. Walpole, R. H. Myers, S. L. Myers, and K. Ye, Probability and Statistics for Engineers and Scientists, Boston, MA: Pearson Education, 2007.

[46] M. Friedman, “A Comparison of Alternative Tests of Significance for the Problem of m Rankings,” *Annals of Mathematical Statistics*, vol. 11, no. 1, pp. 86–92, Mar. 1940.

[47] C. Gao, H. Wang, L. Zhai, Y. Gao, and S. Yi, “An Energy-Aware Ant Colony Algorithm for Network-Aware Virtual Machine Placement in Cloud Computing,” in *Proc. IEEE 22nd International Conference on Parallel and Distributed Systems (ICPADS)*, Wuhan, China, pp. 669-676, Dec. 2016.

[48] Q. Zheng, R. Li, X. Li, N. Shah, J. Zhang, F. Tian, K. M. Chao, and J. Li, “Virtual machine consolidated placement based on multi-objective biogeography-based optimization,” *Future Generation Computer Systems,* vol. 54, pp. 95-122, Jan. 2016.

[49] C. Wei, Z. Hu and Y. Wang, “Exact algorithms for energy-efficient virtual machine placement in data centers,” *Future Generation Computer Systems*, vol. 106, pp. 77-91, May. 2020.

[50] L. Zhao, L. Lu, Z. Jin, and C. Yu, “Online Virtual Machine Placement for Increasing Cloud Provider’s Revenue,” *IEEE Transactions on Services Computing,* vol. 10, no. 2, pp.273-285, Apr. 2017.

[51] J. Wang, N. Ganganath, C. Cheng and C. Tse, "Bio-Inspired Heuristics for VM Consolidation in Cloud Data Centers," *IEEE Systems Journal*, vol. 14, no. 1, pp. 152-163, Mar. 2020.

[52] D. Zhao, J. Zhou and K. Li, "An Energy-Aware Algorithm for Virtual Machine Placement in Cloud Computing," *IEEE Access*, vol. 7, pp. 55659-55668, Apr. 2019.

[53] Q. Wu, F. Ishikawa, Q. Zhu and Y. Xia, "Energy and Migration Cost-Aware Dynamic Virtual Machine Consolidation in Heterogeneous Cloud Datacenters," *IEEE Transactions on Services Computing*, vol. 12, no. 4, pp. 550-563, Aug. 2019.

[54] W. El-Ashmawi and D. Elminaam,"A modified squirrel search algorithm based on improved best fit heuristic and operator strategy for bin packing problem," *Applied Soft Computing*, Vol. 82, Sep. 2019.

[55] S. Wang, A. Zhou, C. H. Hsu, X. Xiao, and F. Yang, “Provision of Data-Intensive Services Through Energy- and QoS-Aware Virtual Machine Placement in National Cloud Data Centers,” *IEEE Transactions on Emerging Topics in Computing,* vol. 4, no. 2, pp. 290-300, Jun. 2016.

[56] J. Del Ser, E. Osaba, D. Molina, X. S. Yang, S. Salcedo-Sanz, D. Camacho, S. Das, P. N. Suganthan, C. A. Coello Coello, and F. Herrera, “Bio-Inspired Computation: Where We Stand and What’s Next,” *Swarm and Evolutionary Computation*, vol. 48, pp. 220-250, Aug. 2019.