Future Generation Computer Systems 102 (2020) 789–809

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Energy-efficient and quality-aware VM consolidation method

Zhihua Li [a,c,*], Xinrong Yu [a], Lei Yu [b], Shujie Guo [a,c], Victor Chang [d]

[a] Department of Computer Science and Technology, School of Internet of Things Engineering, Jiangnan University, Jiangsu Wuxi 214122, China
[b] School of Computer Science, College of Computing, Georgia Institute of Technology, Atlanta, GA, 30332, USA
[c] Jiangsu Provincial Engineering Laboratory of Pattern Recognition and Computational Intelligence, Jiangnan University, Jiangsu Wuxi 214122, China
[d] School of Computing, Engineering and Digital Technologies, Teesside University, Middlesbrough, UK

HIGHLIGHTS

- First, we note that VM placement is an authentic combination optimization issue with multiple resource constraint.
- Then, the probable mappings between VMs and PMs are abstracted as a piece of limited search space, and which corresponds to a population of heuristic evolutionary algorithm. Each individual of population is identical to a real mapping between VMs and PMs during a cycle of VMs consolidation.
- Next, we define a combination optimization model for handling VM placement to achieve the optimal mapping between VMs and PMs in the search space. The solution of the optimization model is performed by an improved heuristic evolutionary algorithm to guarantee the globally optimal results, namely, the optimum VM placement scheme.
- At last, the proposed EQ-VMC method integrates sub-algorithms on host overloading detection, VM selection and under-loaded host detection for VMs consolidation. Comparison and validation are performed using the CloudSim toolkit. The experimental results show that the presented EQ-VMC method is promising in degrading energy consumption and host overloading risk, as well as in improving QoS. Thereby its effectiveness and efficiency have been validated.

ARTICLE INFO

Article history:
Received 13 October 2018
Received in revised form 18 June 2019
Accepted 5 August 2019
Available online 12 August 2019

Keywords:
Virtual machine placement scheme
Optimization model
Discrete DE algorithm
Virtual machine consolidation

ABSTRACT

To improve resource utilization and energy efficiency, cloud data centers use virtual machine (VM) consolidation to consolidate VMs to fewer physical machines (PMs) through live VM migration. However, improper VM placement may cause frequent VM migrations and constant on–off switching of PMs, which results in lower service quality and increased energy consumption. In this paper, we address this problem by proposing an effective and efficient VM consolidation approach called EQ-VMC, which has the goal of optimizing energy efficiency and service quality. In our approach, a discrete differential evolution algorithm is developed to search for the global optimum solution for VM placement. By integrating this solution with a set of algorithms proposed for effective host overload detection, VM selection, and under-loaded host detection, EQ-VMC effectively reduces energy consumption and improves quality of service (QoS). Extensive simulation demonstrates its effectiveness and shows its superiority to previous VM consolidation methods.

1. Introduction

Virtual machine (VM) consolidation is a critical mechanism in cloud computing which migrates VMs to fewer running physical machines (PMs), in order to improve energy efficiency and resource utilization. However, improper VM placement during VM consolidation may further result in frequent live VM migration and constant on–off switching of PMs, leading in turn to serious quality of service (QoS) degradation and resource overhead. Thus, an effective algorithm is crucial for efficient VM consolidation. On the other hand, this is a very challenging issue since it involves several different types of resource factors, such as CPU, memory, network bandwidth, and disk I/O, and VM workloads have dynamic and unpredictable resource demands.

The VM consolidation problem has received significant attention in recent years. Many methods [1–11] have been proposed to address several aspects involved in VM consolidation, including host overload detection, VM placement ,VM selection, and under-loaded host detection. Methods [1–3] determine hotspots by comparing the current resource utilization measurements with the given thresholds. Threshold-based methods, however, cannot adapt to dynamic resource utilization and demand uncertainty. A number of works on VM consolidation [4,5,12–15] have analyzed

* Corresponding author.
E-mail address: zhli@jiangnan.edu.cn (Z. Li).

https://doi.org/10.1016/j.future.2019.08.004
0167-739X/

the factors of VM resource demands and PM workload in data centers and have proposed statistical methods to predict these factors in order to perform VM migration. Many intelligence-related VM consolidation methods [3,12,16–25] also have been developed, but they easily get trapped in local optima, and it is difficult to obtain an ideal balance between energy consumption, resource utilization, and QoS. Some VM consolidation methods [6–10,16] use heuristic algorithms to search for the solution of VM placement that optimally reduces energy consumption but they often suffer from premature convergence, which leads to sub-optimal solutions.

In this paper, we present an energy-efficient and quality-aware VM consolidation (EQ-VMC) method. The EQ-VMC method is a heuristic evolutionary VM consolidation method that seeks to minimize the energy consumption of running PMs while ensuring the lowest possible overloading risk of PM resources via a dynamic optimization model for VM placement. In our approach, an improved discrete differential evolution (discrete DE) algorithm is developed, which searches for the global optimization solution for VM placement of migrated VMs. This algorithm regards all mappings between VMs and PMs as a population and uses heuristic evolutionary to obtain optimal VM placement. At the same time, the developed discrete DE algorithm accelerates the process of finding the global optimization solution by employing the strategy of multi-evolution routes. Additionally, we propose a set of methods for different phases in VM consolidation, including host overload detection, VM selection, and under-loaded host detection. Our major contributions can be summarized as follows:

(1) First, we note that VM placement is a challenging combination optimization issue with multiple resource constraints;

(2) Second, the probable mappings between VMs and PMs are abstracted as a limited search space which corresponds to the population of a heuristic evolutionary algorithm. Each individual of the population is identical to a real mapping between VMs and PMs during a single cycle of VM consolidation;

(3) Next, we define a combination optimization model for handling VM placement to achieve the optimal mapping between VMs and PMs in the search space. The solution of the optimization model is performed by using an improved discrete differential evolution algorithm to guarantee the global optimization results—that is, the optimum VM placement scheme;

(4) Finally, the proposed EQ-VMC method integrates sub-algorithms for host overload detection, VM selection and under-loaded host detection for VM consolidation. Comparison and validation are performed utilizing the CloudSim toolkit. The experimental results show that the presented EQ-VMC method shows promise for decreasing energy consumption and host overloading risk, as well as improving QoS. Thus, the effectiveness and efficiency of EQ-VMC method have been validated.

The rest of this paper is organized as follows. The related works are introduced in Section 2. In Section 3, we describe the optimization model and scheme for VM placement. In Section 4, we derive the framework of the EQ-VMC method and propose the overview and the detailed design of it. Extensive experiment results are given in Section 5, followed by the concluding remarks and future works in Section 6.

2. Related works

A VM consolidation scheme should determine where the VMs migrate from and migrate to, as well as which PMs can be turned off; in other words, it should resolve the issue of identifying the source and destination PMs for live VM migration. Many works [2,3,12,16–18,20–24,26] carry out this issue from various perspectives, such as VM placement [3,16–18], host overload detection [2,3,12,18,20–22], and VM migration selection [3,23,24]. In this paper, we focus on heuristic-evolution-based VM consolidation methods, thus we mainly discuss the related topics of VM placement and VM consolidation methods.

2.1. VM placement

Efficient VM placement is critical for VM consolidation, due to the fact that an inappropriate VM placement scheme easily reduces resource utilization and increases energy consumption, possibly leading to risk of new host overloading. The core of VM placement addresses the issue of "where to" for live VM migration, and it has been characterized as approximating a bin packing problem [17,27–29]. Mishra and Sahoo [28] handled this issue as a multi-dimensional bin packing problem, optimizing the mapping between VMs and PMs as specific optimization objectives by using an improved genetic algorithm [27], first-fit decreasing (FFD) algorithm [27], and other intelligence optimization algorithms. Although both Kaaouache et al. [27] and Mishra and Sahoo [28] addressed improving QoS and resource utilization by optimizing the mapping, both of them did not regard the dynamic nature of the up-allocating VMs and PMs, which easily results in frequent VM migrations and constant on–off switching of PMs. Best fit decreasing (BFD) [30] is an effectively heuristic algorithm employed to resolve this packing issue. A Power-Aware BFD (PABFD) algorithm [17] based on BFD is presented. The PABFD algorithm first performs the unallocated VMs in descending order based on their CPU resource request and then allocates each VM to the destination PM according to this order. Each VM deploys on the PM with creates a minimal increase in energy consumption. However, both BFD and PABFD algorithms do not consider the workload changes of the destination PM after VM placement, which may induce a new risk of host overloading, thus not ensuring QoS. J. A. Aroca et al. [29] performed competitive ratio analysis on approximate solutions for the VM placement issue with restrictions on the number of VMs and PMs. Melhem et al. [31] proposed a Markov prediction model for VM placement in live VM migration to determine the set of candidate destination PMs that would be able to receive the migrated VMs in a way that avoids their VM migration in the near future. However, they did not consider how to determine the under-loaded, over-loaded, or normal-loaded status, which is also important for VM consolidation.

2.2. Heuristic VM consolidation

VM consolidation primarily includes host overload detection, VM migration selection, VM placement, and running PMs shrinking [2,9]. Due to the complexity of VM consolidation, it [9] has traditionally been divided into several sub-problems, the task then being conducted by handling and integrating each of the sub-problems. When using competitive ratio analysis, this type of method is very effective in practice in terms of viewpoints [32], even though it is unable to guarantee optimal results theoretically. So far, a large number of studies [1–16] have addressed the VM consolidation involved in different phases, and heuristic algorithms have been implemented for VM consolidation owing to their outstanding performance in resolving complex multi-objective optimization models. Several researches [6,7,9,10,16] have explored this issue, with benchmark performance. Telenyk et al. [10] aimed to improve QoS with respect to reducing energy consumption and proposed to minimize the imbalance between each resource of the PMs with a simulated annealing algorithm (SA-VMC) to find the optimal VM placement. However, this method does not consider stochastic demands, which can result in new overloading risk and further load imbalance. Generally speaking, fewer running PMs indicates lower energy consumption in data centers. Based on this idea, Hallawi et al. [7] developed a multi-objective optimization model, minimizing both the number of running PMs and total resource wastage. The genetic algorithm COFFGA was proposed to perform VM consolidation, using chromosome encoding to represent the VM placement order

and obtaining the best placement order via evolution. However, COFFGA employs an FFD algorithm for VM placement, which tends to result in insufficient reserved resources in PMs, and it suffers from reduction of QoS and improper optimization objectives. Farahnakian et al. [6] proposed to minimize the total number of running PMs with the goal of reducing the frequency of VM migration (VMMs) to improve QoS. They proposed an algorithm called Ant Colony System (ACS) to resolve the optimization model. In ACS, a set of tuples, T, is created, where each tuple consists of three elements: the source PM, the VM to be migrated, and the destination PM. Each tuple represents a sketch of VM migration. ACS finds the best tuple set from the total probable VM migrations via the ant colony algorithm. The proposed ACS-VMC decreases energy consumption by consolidating VMs and reducing the total number of running PMs. But the objective of minimizing VMMs easily causes new host overloading risk, further increasing SLA violations. Gao et al. [16] proposed VMPACS, which utilizes an ACS algorithm to address VM consolidation with the goal of directly reducing energy consumption and minimizing resource wastage. VMPACS searches for the optimum VM placement which balances the available resources on each PM along varying dimensions. The method efficiently and simultaneously minimizes total resource wastage and energy consumption. Although VMPACS is superior to the aforementioned algorithms in reducing energy consumption, it does not regard how to reserve resources of PMs to guarantee QoS. Li et al. [9] proposed EC-VMC, which searches the optimal mapping between VMs and PMs for live VM migration to minimize energy consumption and VMMs by simulating Artificial Bee Colony (ABC) foraging behavior. EC-VMC established a multi-objective optimization model and converts multiple objectives into a single objective through ξ constraints and resolves this single-objective optimization problem by using an ABC-like algorithm. Objectively speaking, although the ABC algorithm can achieve global optimization, but it is unhelpful in resolving multi-peak optimization problems. This is a deficiency in the inherent nature of the EC-VMC algorithm.

The proposed studies are concerned with research on the VM placement model and the related VM consolidation method. The proposed model focuses on the optimal balance between energy consumption and QoS. Accordingly, we present the EQ-VMC method. The major differences from the previous works are as follows:

(1) First, the presented VM placement model focuses on the balance between energy consumption and PM workload stability during ongoing VM consolidation;

(2) Next, searching the final optimum results in the solution space (e.g., population) is performed by the improved discrete differential evolution algorithm, and it not only guarantees the global optimization result but accelerate the evolutionary process;

(3) At last, by integrating several sub-algorithms with the discrete DE-based VM placement algorithm, EQ-VMC globally addresses the "where from and where to" issue of live VM migration, thereby fundamentally guaranteeing the reliability of the results. The extensive experimental results demonstrate that the presented EQ-VMC method efficiently reduces energy consumption and host overloading risks while effectively guaranteeing QoS.

3. System model and problem formulation

3.1. Data center model

Suppose a data center consists of a set of PMs denoted by $H = \{h_1, h_2, \ldots, h_j, \ldots, h_n\}$, and the VMs are represented as $V = \{v_1, v_2, \ldots, v_i, \ldots, v_m\}$. The deployment relation between VMs and PMs is expressed as a mapping matrix $D_{m\times n} = \left(d_1^T, d_2^T, \ldots, d_i^T, \ldots, d_m^T\right)^T$, in which each vector d_i represents a mapping relation between the virtual machine v_i to all PMs in data centers. The vector d_i is called the deployment vector for the virtual machine v_i. If the virtual machine v_i is deployed on the physical machine h_j, then the jth element in the vector $d_{i,j} = 1$, otherwise $d_{i,j} = 0$. Because each VM can only be allocated on a single PM, the deployment vector d_i satisfies $\sum_{j=1}^{n} d_{i,j} = 1$, which is a constraint condition for the VM placement.

For each resource, the configured resource capacity of virtual machine v_i is represented by r_i^{res},where $res \in \{cpu, mem, band\}$. Each type of resource capacity on the physical machine h_j is denoted as c_j^{res}. The utilization of each resource on a PM is computed according to (1),

$$u_j^{res} = \sum_{v_i \in V_j} a_i^{res} / c_j^{res} \tag{1}$$

where V_j denotes the set of VMs deployed on the physical machine h_j, and a_i^{res} represents the real resource utilization on virtual machine v_i in the PM.

As for a physical machine h_j, the total usage of all deployed VMs for a specific type of resource cannot be more than the resource capacity of those VMs, namely, $a_i^{res} < r_i^{res}$. So, for the destination PM, the total requested resources from the deployed VMs must satisfy the following constraint condition [3,5,6,9].

$$\sum_{v_i \in V_j} r_i^{res} < c_j^{res} \tag{2}$$

3.2. Energy consumption estimation

Due to differences in the PMs' hardware heterogeneity and running status, the energy consumption of an identical VM deployed on different PMs will also be different. We follow the energy consumption model of [9] that characterizes the relationship between CPU utilization and energy consumption. It states that, given p intervals $\{[0, 1/p), [1/p, 2/p), \ldots, [(p-1)/p, 1]\}$ of the average CPU resource utilization, the PM energy consumption linearly increases with the CPU resource utilization ratio in each interval. Thus, the energy consumption of PMs can be estimated as shown in formula (3).

$$PM_j^{power}\left(u_j^{cpu}\right) = \begin{cases} \lambda_1 \cdot u_j^{cpu} + \eta_1, & 0 \le u_j^{cpu} < 1/p \\ \lambda_2 \cdot u_j^{cpu} + \eta_2, & 1/p \le u_j^{cpu} < 2/p \\ \vdots & \vdots \\ \lambda_p \cdot u_j^{cpu} + \eta_p, & (p-1)/p \le u_j^{cpu} \le 1 \end{cases} \tag{3}$$

where $\lambda_i (i = 1, 2, \ldots, p)$ is the slope of the linear function of each power interval, $\eta_i (i = 1, 2, \ldots, p)$ is energy consumption at different load levels in watts with respect to the hardware heterogeneity of PMs (e.g., Table 3 in Section 5.1).

Based on formula (3), the energy consumption for a PM in a certain period of time can be estimated as

$$EC\left(h_j\right) = \gamma_j \cdot \int_{t_0}^{t_1} PM_j^{power}\left(u_j^{cpu}(t)\right) dt \tag{4}$$

where $\gamma_j \in \{0, 1\}$. If $\gamma_j = 1$ then the physical machine h_j is running; if $\gamma_j = 0$, the physical machine h_j is in sleep mode.

3.3. Host overloading probability estimation

Host overloading probability reflects the overloading risk of PMs given the current deployed VMs. When the resource utilization gets high, the uncertainty of workload will increase the host overloading risk. In this paper,we follow the approach of [33].

It characterizes the stochastic variation of VM resource requests with a normal distribution, and then estimates the distribution of each resource usage on PMs. The host overloading probability is determined as

$$P_{over}\left(h_j\right) = 1 - \prod_{res \in \{cpu, mem, band\}} \Pr_{res}\left(\sum_{v_i \in V_j} r_i^{res} < c_j^{res}\right) \tag{5}$$

where $\Pr_{res}$ is a probability distribution function of the usage of various resources on a PM.

3.4. Problem formulation for VM placement

In terms of the aforementioned energy consumption estimation for data centers and analysis of host overloading risk, the optimization problem for VM placement in datacenters can be defined as shown in formula (6). In fact, formula (6) attempts to realize the optimization objective, obtaining the optimum mapping relationship between VMs and PMs by taking the mathematical expectation of energy consumption of PMs with the lowest overloading risk.

$$\begin{aligned} \operatorname{argmin} f(D) = & \sum_{\substack{f: vm_i \to h_j \\ f \in D}} EC(h_j) \cdot e^{\alpha \cdot P_{over}(h_j)} \\ s.t. \quad & \sum_{j=1}^{n} d_{i,j} = 1, i = 1, 2, \ldots, m \\ & c_{j,res} \cdot u_{j,res} + \sum_{i=1}^{m} r_{i,res} \cdot d_{i,j} < c_{j,res}, \\ & j = 1, 2, \ldots, n, res \in \{cpu, mem, band\} \end{aligned} \tag{6}$$

where the host overloading probability is $P_{over}(h_j) \in [0, 1]$, the function $e^{\alpha \cdot P_{over}(h_j)}$ adjusts the effect of the host overloading probability. When the overloading risk increases during VM placement, the importance of host overloading probability rises exponentially. This behavior has the effect of enhancing the sensitivity of the model to host overloading probability. Further, the parameter α is used to control the impact of overloading probability on the optimization objective. The greater the value of α is, the more sensitive the destination PMs are to the host overloading probability. Unfortunately, when the host overloading probability tends towards 0, the identically migrated VM will often then be able to migrate to multiple under-loaded destination hosts. Under such circumstances, VM placement can be determined according to energy consumption. It can be clearly seen that formula (6) can fully realize the optimization objectives of minimizing energy consumption of running PMs with minimal overloading risk.

4. Our VM consolidation approach

In this section, we present our VM consolidation approach. It consists of the multi-resource host overload detection (MHOD) algorithm, QoS-aware VM selection (QVMS) algorithm, discrete DE based VM placement (Discrete DEVMP) algorithm, and underloaded hosts detection (ULHD) algorithm. These four algorithms cooperate and integrate with each other for live VM migration with guaranteed QoS, improving resource utilization and reducing energy consumption. We describe them one by one in what follows.

Additionally, for the clarity, all used notations and their meanings are listed in Table 1.

4.1. VM placement

To resolve the VM placement problem formulated in Section 3.4, we present an improved Discrete DE algorithm below.

4.1.1. Discrete DE algorithm

The differential evolution (DE) algorithm [34] consists of three operations – mutation, crossover and selection – and mainly handles continuous variables. The population of the traditional DE algorithm consists of several vectors, and their values in each dimension are continuous. However, in this paper, we have to process discrete variables and thus need to discretize the typical DE algorithm. Given a population of S individuals denoted as $X^t = \left\{D_0^t, D_1^t, \ldots, D_k^t, \ldots, D_S^t\right\}$, in which D_k^t is a matrix and t is the generation of population, the detailed processes are as follows:

4.1.1.1. Mutation. Arbitrarily select three individuals $D_{r_1}, D_{r_2}, D_{r_3}$ from the population X^m, in which D_k^m is calculated according to formula (7),

$$D_k^m = D_{r1} + F \cdot (D_{r2} - D_{r3}), \quad k = 1, 2, \ldots, S \tag{7}$$

where $F \in [0, 2]$ is a scalar factor that adjusts the impact to the difference vector. The ith row in the difference matrix D_Δ is determined according to formula (8),

$$d_{\Delta,i} = d_{r2,i} - d_{r3,i} = \begin{cases} d_{r2,i}, & d_{r2,i} \neq d_{r3,i} \\ \vec{0}, & others \end{cases} \quad i = 1, 2, \ldots, m \tag{8}$$

where $d_{\Delta,i}, d_{r2,i}, d_{r3,i}$ represent the ith row corresponding to matrix D_Δ, D_{r2}, D_{r3} respectively. Formula (8) obtains the result (e.g., the row in the difference matrix) by determining whether the deployment vectors at the locations corresponding to the two matrices are equal. Further, the mutation individual D_k^m is calculated by formula (9).

$$\begin{aligned} d_{k,i}^m &= d_{r1,i} + F \cdot d_{\Delta,i} \\ &= \begin{cases} d_{\Delta,i}, & 0 < randf < F \ \& \ d_{\Delta,i} \neq \vec{0} \\ d_{r1,i}, & others \end{cases} \quad i = 1, 2, \ldots, m \end{aligned} \tag{9}$$

where $d_{k,i}^m$ is the ith row in matrix D_k^m, $randf$ is a random variable, and $randf \in [0, 2]$.

4.1.1.2. Crossover. The cross-calculation is determined by formula (10).

$$d_{k,i}^c = \begin{cases} d_i^m, & randc \leq CR \text{ or } i = randi \\ d_i, & randc > CR \ \& \ i \neq randi \end{cases} \quad i = 1, 2, \ldots, m \tag{10}$$

where $d_{k,i}^c$ is the ith row in the cross-mapping matrix D_k^c, $CR \in [0, 1]$ and CR is a cross constant, $randc$ is a random variable with $randc \in [0, 1]$, $randi$ is random data in the sequence $\{1, 2, \ldots, m\}$.

4.1.1.3. Selection. The selection operation is performed by comparing the value of f one-by-one between the original individuals in the initial population and corresponding cross individuals, which select individuals with small values of f that join next-generation populations. The detailed selection process is shown in formula (11).

$$D_k^{t+1} = \begin{cases} D_k^c, & f\left(D_k^c\right) < f\left(D_k^t\right) \\ D_k^t, & others \end{cases}, k = 1, 2, \ldots, S \tag{11}$$

4.1.2. Discrete DE-based VM placement algorithm

The mappings between PMs and VMs in data centers form a limited search space, and they are abstracted as a population corresponding to the X^t in Section 4.1.1,where each individual D_k^t in the population represents one of the probable mapping matrices between VMs and PMs. The ith row in D_k^t, $d_{k,i}^t$ is the deployment vector of the virtual machine v_i, which essentially refers to the deployment map on all running PMs for virtual machine v_i. The corresponding $d_{k,i,j}^t$ represents the deployment component of the deployment vector $d_{k,i}^t$. Here, when the individual obtained by the mutation, crossover, and selection operations by the Discrete DE algorithm is the final optimum mapping relationship between

Table 1

.

v_i	The ith VM
V	The set of VMs in datacentres
V_{mig}	The set of migrated VMs
h_j	The jth PM
H	The set of PMs in data center
H_{active}	The set of active PMs
H_{over}	The set of overloading PMs
H_{mig}	The set of new destination PMs
H_s	The set of PMs need under-loaded detection
H_{sp}	The set of PMs are suitable to place VMs from under-loaded PM
m	The number of VMs
n	The number of PMs
$D_{m\times n}$	A mapping matrix denotes the deployment relation between VMs and PMs
d_i	The deployment vector for v_i
$d_{i,j}$	The deployment component for v_i to h_j
res	The resource types of PMs and the set is cpu, mem, $band$
r_i^{res}	The resource demand of v_i
a_j^{res}	The real amount of resource allocated to the VMs on h_j
c_j^{res}	The resource capacity of h_j
u_j^{res}	The resource utilization of h_j
p	The number of intervals on CPU utilization
λ_i	The slope of the linear function of each power interval
η_i	The intercept of the linear function of each power interval
γ_j	The flag whether h_j is active or not
Pr_{res}	The normal distribution function of various resource usage on a host
α	The weight of overloading probability in optimization model
ω	The safe parameter to adjust threshold
X^t	The population of t-th generation
S	The size of population
D_k^t	The kth individual in population
D_Δ	The difference matrix
D_{r1}, D_{r2}, D_{r3}	Three random selected individuals from population
D_k^m	The kth mutant individual in population
, $d_{k,i}^m$, $d_{k,i,j}^m$	The deployment vector and deployment component in D_k^m
D_k^c	The kth crossover individual in population
$d_{k,i}^c$, $d_{k,i,j}^c$	The deployment vector and deployment component in D_k^c
F	A scalar factor and its value range is $[0, 2]$
$randf$	A random variable and its value range is $[0, 2]$
CR	A cross constant and its value range is $[0, 1]$
$randc$	A random variables and its value range is $[0, 1]$
$randi$	A random data in the sequence $\{1, 2, \ldots, m\}$
$HMatrix^t(i, j)$	An element in heuristic information matrix of the t-th generation
β	The relative weight of two factors for initial heuristic values
ρ	The parameter adjusts the proportion between previous generation heuristic information and current heuristic information
Δc_{j+i}^{cpu}	The remain CPU resource of h_j after hosting v_i on it
u_{j+i}^{res}	The resource utilization of h_j after hosting v_i on it
v_{mig}	The migrated VMs
a_{mig}^{mem}	The allocated memory for v_{mig}
$\mathrm{Pr}_{res}^{-v_{mig}}$	The normal distribution of a resource usage in which hosts exclude v_{mig}
EXP	The rate of exploratory evolution and its value range is $[0, 1]$
$rand_{\exp}$	A random value and its value range is $[0, 1]$
$\xi_j^{violate}$	The SLAV duration resulting from overloaded CPU resources for h_j
ξ_j	The running time of h_j
R_i^{mig}	The size of the unsatisfied demand for CPU resources as a result of v_i migration
R_i	The size of the demand for CPU resources from v_i

VMs and PMs, then it is called the VM placement scheme in this paper.

In this section, we propose an improved Discrete DE algorithm to solve the VM placement problem given in Eq. (6). In the improved Discrete DE algorithm, we additionally employ a heuristic information matrix to accelerate the evolutionary process. The concept diagram for the discrete DE algorithm is shown in Fig. 1.

In Fig. 1, given X^t which generates the population of the ith generation, the individuals are randomly separated into two evolution routes, X_{evolve}^t and X_{explore}^t. The X_{evolve}^t route depends on the heuristic information matrix; X_{explore}^t directly explores the results in the search space. The heuristic information matrix *HMatrix* is updated at the same time as the current generated population, thereby speeding up the evolution and avoiding premature local optimization. Finally, this yields a new population X^{t+1}.

Next we will discuss the following details: updating the heuristic information matrix and the evolution routes of X_{evolve}^t and X_{explore}^t.

4.1.2.1. Updating the heuristic information matrix. In Fig. 1, each element in the heuristic information matrix has a one-to-one correspondence with each deployment component. For example, supposing $HMatrix^t(i, j)$ records the heuristic information of the deployed virtual machine vm_i on physical machine h_j in the current population X^t. At this time, if the virtual machine v_i can be deployed on the physical machine h_j, the initial value of $HMatrix^0(i, j)$ can be computed with formula (12); otherwise, the initial value is 0.

$$HMatrix^0(i,j) = \beta \cdot (\Delta c_{j+i}^{cpu} / \Delta PM_{j+i}^{power}) + (1-\beta) \cdot 1/\mathrm{std}(u_{j+i}^{cpu}, u_{j+i}^{mem}, u_{j+i}^{band}) \tag{12}$$

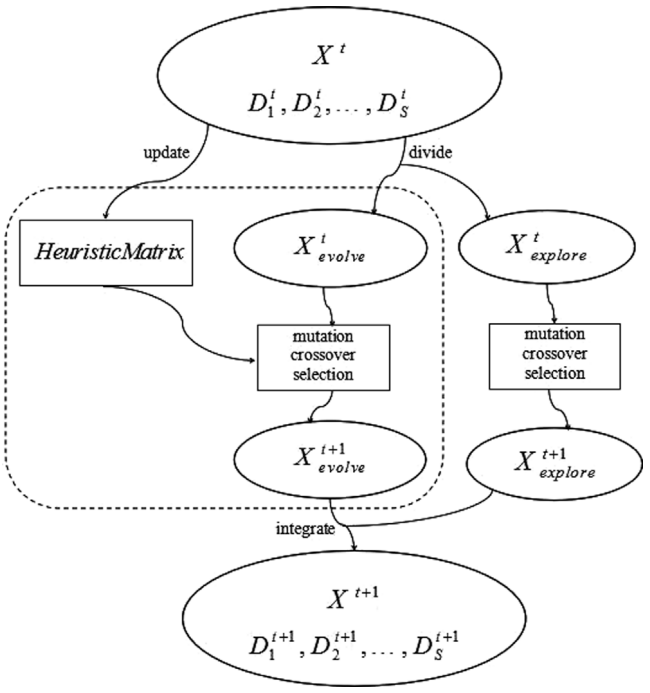


Fig. 1. The concept diagram of discrete DE algorithm.

where $\Delta c_{j+i}^{cpu} = c_j^{cpu} - (a_j^{cpu} + r_i^{cpu})$ is the CPU resource remaining on h_j after hosting v_i on it; $\Delta PM_{j+i}^{power} = PM_j^{power}(1) - PM_j^{power}((\sum_{v_i \in V_j} a_i^{cpu} + r_i^{cpu})/c_j^{cpu})$ is the remaining power of h_j; the large value $\Delta c_{j+i}^{cpu}/\Delta PM_{j+i}^{power}$ represents the higher cost performance of h_j for other VMs. std$(u_{j+i}^{cpu}, u_{j+i}^{mem}, u_{j+i}^{band})$ is the standard deviation of $h_j^{'s}$ resource utilization after hosting v_i, the large value $1/\text{std}(u_{j+i}^{cpu}, u_{j+i}^{mem}, u_{j+i}^{band})$ represents the more stable resource utilization of h_j after finishing VM placement. β is the weight of these two objectives.

Each D_k^t in population X^t has a corresponding value $f(D_k^t)$ according to formula (6). The range of f is determined by the deployment vector $d_{k,i}^t$ (*e.g.*, $i = 1, 2, \ldots, m$) in the current mapping matrix D_k^t. Usually, the smaller the value of f, the more favorable the VM placement scheme; conversely, the larger the value of f, the worse the VM placement scheme. Further, the priority level of each deployment component is quantified by $1/f(D_k^t)$. In addition, $\sum_{k=1}^{S}\left((1/f(D_k^t)) \cdot d_{k,i,j}^t\right)/\sum_{k=1}^{S} d_{k,i,j}^t$ tracks the average priority level of the deployment component in the population. At the same time, the heuristic information matrix is updated according to formula (13),

$$\begin{aligned} &HMatrix^{t+1}(i,j) \\ &= \rho \cdot HMatrix^t(i,j) + (1-\rho)\left(\sum_{k=1}^{S} 1/f(D_k^{t+1}) \cdot D_{k,i,j}^{t+1}\right) \Big/ \\ &\sum_{k=1}^{S} D_{k,i,j}^{t+1}, D_k^{t+1} \in X^{t+1} \end{aligned} \tag{13}$$

where $\rho \in [0, 1]$ weights the previous generation of heuristic information and current heuristic information. In terms of the above element value of heuristic matrices and their update process, it can be seen that a deployment component satisfying $d_{k,i,j}^t = 1$ will more frequently appear in D_k^t with a small f value, and the heuristic information value $HMatrix^t(i,j)$ of the corresponding deployment component will get bigger. Namely, the element value of the heuristic information matrix reflects the priority of the corresponding deployment component; that is, a larger element value for the heuristic information matrix means that its corresponding deployment component contributes more weight to the optimization objectives.

4.1.2.2. The evolution of X_{evolve}^t. Under the guidance of the heuristic information matrix, the mutation operation is discussed first. The mutation operation is meant to yield more ideal deployment components for the deployment vector. Suppose that three mapping relations D_{r1}, D_{r2}, D_{r3} in population X^t are randomly selected. The deployment vector $d_{\Delta,i}$ of difference matrix D_Δ is computed by formula (14),

$$\begin{aligned} &d_{\Delta,i} \\ &= d_{r2,i} - d_{r3,i} \\ &= \begin{cases} d_{r2,i}, & \sum_{j=1}^{n} d_{r2,i,j} \cdot HMatrix^t(i,j) \geq \sum_{j=1}^{n} d_{r3,i,j} \cdot HMatrix^t(i,j) \\ \vec{0}, & others. \quad i = 1, 2, \ldots, m \end{cases} \end{aligned} \tag{14}$$

where $d_{r2,i}$, $d_{r3,i}$ denote the deployment vectors of D_{r2}, D_{r3}, respectively, and $d_{r2,i,j}$, $d_{r3,i,j}$ are the corresponding deployment components. In fact, formula (14) performs the subtraction by comparing the heuristic information of the two deployment vectors. However, when the heuristic information is relatively small, the vector $d_{\Delta,i}$ equals "$\vec{0}$". The differential matrix obtained by formula (14) achieves a more ideal deployment vector. Additionally, since the obtained differential matrix contains more heuristic information, such methods can guide the subsequent evolutionary route to quickly find the optimum VM placement scheme. Finally, the deployment vector $d_{k,i}^m$ of the mutant individual D_k^m is obtained by formula (9).

Next, the crossover operation handles the mapping relation between the original mapping and the mutated mapping between PMs and VMs. The individuals updated via crossover may have a mapping relation that contains a slice of superior deployment components identified in the heuristic matrix. The crossover D_k^c of the original individuals and mutated individuals is generated by formula (10). It can be seen that the crossover operation of Eq. (10) randomly selects the deployment components of mutated individuals to cross. However, when selecting the deployment components of the cross individual one by one, it is necessary to constrain each deployment vector $d_{k,i}^c$ of the crossover individual D_k^c according to the PMs' resource constraints under the presented optimization model, so as to ensure that the crossover individual will become a VM placement scheme that meets the PMs' resource constraints. When it does not satisfy the PMs' resource constraints, the deployment components of the crossover individual are defined by formula (15).

$$d_{k,i,j}^c = \begin{cases} 1, & \max_j\{HMatrix^t(i,j)\} \ \& \ \sum_{v_i \in V_j}\left(d_{k,i,j}^m \cdot r_i^{res}\right) < c_j^{res} \\ 0, & otherwise \end{cases} \tag{15}$$

Formula (15) aims to find the most appropriate deployment component—namely, the ith deployment vector $d_{k,i}^c$ of the crossover individual D_k^c.

4.1.2.3. The evolution of $X_{explore}^t$. If the individual follows the exploratory evolution route, the corresponding mutated individuals are obtained directly according to formulas (8) and (9). This operation allows for fully exploring other feasible mutated individuals in the search space. Similarly, the crossover mapping is obtained by exploring the original mapping relation of some deployment

components, thus possibly avoiding the evolution algorithm getting trapped in a local optimum. When the deployment vector $d^c_{k,i}$ does not satisfy the PMs' resource constraints, recalculate each deployment component of $d^c_{k,i}$ with formula (16).

$$d^c_{k,i,j} = \begin{cases} 1, & \sum_{v_i \in V_j} \left(d^m_{k,i,j} \cdot r^{res}_i\right) < c^{res}_j, \ wherein\ random\ j \\ 0 & otherwise \end{cases} \tag{16}$$

where a deployment component is randomly selected in line with the PMs' resource constraints during VM placement, which fully embodies the generality of the exploratory evolution route.

4.1.2.4. VM placement algorithm. The optimum mapping relationship between VMs and PMs is achieved through mutation, crossover, and selection; and heuristic information matrix updates synchronously by the Discrete DE algorithm, i.e., the VM placement scheme. Based on this, the Discrete DE-based VM Placement (Discrete DEVMP) algorithm is presented. The Discrete DEVMP algorithm is described below.

Algorithm 1: Discrete DE based VM Placement

Input: V, H
Output: $Map_{mig} < host, vm >$
Generate $X^0 = \left\{D^0_0, D^0_1, ..., D^0_S\right\}$, $X^t = X^0$
Use equation (12) init $HMatrix^t$
while $cur\_times < total\_times$
  Update $HMatrix^t$ using equation (13)
  for all D^t_k in X^t
    if $rand_{exp} < EXP$
      Use equation (14),(9) to calculate mutated deployment map D^m_k
      Use equation (10),(15) to calculate cross deployment map D^c_k
    else
      Use equation (8),(9) to calculate mutated deployment map D^m_k
      Use equation (10),(16) to calculate cross deployment map D^c_k
    end if
    $D^{t+1}_k = f(D^t_k) < f(D^c_k) ? D^t_k : D^c_k$
    Add D^{t+1}_k to X^{t+1}
  end for
end while
$D_{optimal} = \arg\min_{D^t_k} f(D^t_k)$
Convert $D_{optimal}$ to $Map_{mig} < host, vm >$

4.2. Host overload detection

First, we use the normal distribution model to fit the history records of PM resource usage and obtain the probability distribution function $\Pr_{res}$ of each resource accordingly. Next, the overloading probability of each resource is obtained depending on $\Pr_{res}$. Finally, the overloading threshold of each resource is determined according to the overloading probability. The overloading threshold is conducted in terms of formula (17),

$$T^{res}_j = 1 - \omega \cdot \Pr_{res}\left(\sum_{v_i \in V_j} r^{res}_i > c^{res}_j\right) \tag{17}$$

where T^{res}_j is the upper boundary for the current utilization of each resource on physical machine h_j. The overloading probability coefficient is an empirical parameter, which requires users to predefine it according to historical experience. When ω gets larger, the overloading threshold of PMs is lowered, so that more resources are reserved for any sudden emergence of requested resources from VMs; and when ω gets smaller, the resource allocation strategy will tend to make full use of the resources of PMs. During host overload detection, any resource utilization exceeding a corresponding threshold value (e.g., $u^{res}_j > T^{res}_j$) is considered an occurrence of host overloading. In data centers, all PMs with overloading risk need to be discovered.

Multi-resource host overload detection (MHOD) algorithm is as follows.

Algorithm 2: MHOD

Input: H_{active}
Output: H_{over}
for all h_j in H_{active}
  if $u^{res}_j > T^{res}_j$
    $H_{over} \leftarrow H_{over} \cup h_j$
  end if
end for

4.3. VM selection

In data centers, usually, it is necessary to migrate some VMs from the overloaded PMs to decrease their overloading risk. Thus, this paper proposes two criteria for performing VM selection: (1) the workload on source PMs gets more stable after VM migration; (2) the time for live VM migration should be as short as possible, so as to minimize its negative influence on QoS. The first criterion can be measured in terms of the changes of overloading probability on source and destination PMs after carrying out VM migration. For the second criterion, the pre-copy mechanism of [35] is used; i.e., the smaller the memory usage, the shorter the migration time, thus effectively reducing the negative impact of VM migration on QoS. The overloading probability of source PMs is estimated with formula (18) after VM migration.

$$P^{-v_{mig}}_{over}\left(h_j\right) = 1 - \prod_{res \in \{cpu, mem, band\}} \Pr^{-v_{mig}}_{res}\left(\sum_{v_i \in V_j} r^{res}_i < c^{res}_j\right), v_{mig} \notin V_j \tag{18}$$

where $\Pr^{-v_{mig}}_{res}$ is the resource utilization of source PM h_j after VM migration. v_{mig} represents the migrated VM.

Furthermore, the two above described criteria for selecting the VMs to be migrated are called the VM selection criteria, quantified according to formula (19),

$$RaP = P^{-v_{mig}}_{over} \cdot \frac{a^{mem}_{mig}}{c^{mem}_j} \tag{19}$$

where a^{mem}_{mig} is the amount of memory occupied by the migrated VM.

For the source PM after VM migration, it is also necessary to continue the overloading risk assessment to confirm whether or not the current source PM has relieved overloading risk. If the current VM migration does not eliminate host overloading risk, it needs to repeatedly perform the aforementioned steps until host overloading risk is removed.

QoS-aware VM Selection (QVMS) algorithm is as follows.

Algorithm 3: QVMS

Input: H_{over}
Output: $V_{migrate}$
for all h_j in H_{over}
  while $u^{res}_j > T^{res}_j$
    $v_m \leftarrow \arg\min_{v_i \in V_j} RaP$
    $V_{mig} \leftarrow V_{mig} \cup v_m$, $V_j \leftarrow V_j - v_m$
  end while
end for

Table 2
The PM instances.

Type	CPU	RAM (GB)
IBM server X3550 M3–1	Intel Xeon X5670 12 cores 2933 Hz	12
IBM server X3550 M3–2	Intel Xeon X5675 12 cores 3067 Hz	16
HP Enterprise ProLiant DL 360 Gen9	Intel Xeon E5-2699 v3 36 cores 2300 Hz	64
HP Enterprise ProLiant DL 360 Gen10	Intel Xeon Platinum 8180 28 cores 2500 Hz	48

Table 3
The energy consumption at different load levels in watts with respect to different PM sent load levels in watts with respect to different PMs.

Utilization (%)	0	10	20	30	40	50	60	70	80	90	100
M3–1	66	107	120	131	143	156	173	191	211	229	247
M3–2	58.4	98	109	118	128	140	153	170	189	205	222
Gen9 (kWh)	45.0	83.7	101	118	133	145	162	188	218	248	276
Gen10 (kWh)	38.7	68.9	82.2	94.6	107	121	138	156	178	210	233

Table 4
The VM instances.

Type	CPU frequency (MIPS)	RAM (GB)
m3.medium	2300×1	3.75
m3.large	2300×2	7.5
m3.xlarge	2300×4	15
c4.large	2300×2	3.75
c4.xlarge	2300×4	7.5
c4.2xlarge	2300×8	15

4.4. Under-loaded hosts detection

During VM consolidation, turning off under-loaded running PMs is useful to reduce energy consumption and improve resource utilization. Therefore, in this section, we address the criteria for under-loaded host detection. During the process of shutting down the under-loaded PMs, the following problems need to be addressed: (1) detection of the under-loaded PMs; (2) migration of all VMs from the under-loaded PM; (3) selection of the destination PM for live VM migration. Obviously, during a cycle of VM consolidation: (1) the under-loaded PMs must be selected from the current running PM set. Because the overloading hosts are in a relatively stable status after VM migration, the previous overloading hosts cannot possibly still be at under-loaded status; (2) those PMs that have been selected as destination PMs in the previous cycle of VM consolidation should also be excluded so that the under-loaded PM candidate set is optimized as $H_s = H_{active} - H_{over} - H_{mig}$. Furthermore, for the under-loaded PM candidate set, the comprehensive utilization of each resource is calculated according to formula (20), and the comprehensive resource utilization is sorted in ascending order, which in turn turns off the low-utilization PMs and migrates all of the VMs on it. The candidate set of destination PMs for the migrated VMs is denoted as $H_{sp} = H_{active} - H_{over}$.

$$u_j = \sqrt{\left(u_j^{cpu}\right)^2 + \left(u_j^{mem}\right)^2 + \left(u_j^{band}\right)^2} \tag{20}$$

Under-loaded hosts detection (ULHD) algorithm is as follows.

Algorithm 4: ULHD
1: Input: H_s, H_{sp}
2: Output: $Map_s < host, vm >$
3: Sort H_s by equation (20)
4: for all h_j in H_s
5: Get D_s from $Discerte\ DEVMP(V_j, H_{sp} - h_j)$
6: if $D_s \neq null$
7: Convert D_s to $map < host, vm >$
8: $H_{sp} \leftarrow H_{sp} - h_j$
9: end if
10: $Map_s < host, vm > \leftarrow Map_s < host, vm > \mathrm{U} map < host, vm >$
11: end for

4.5. VM consolidation method

Naturally, VM consolidation targets to optimize the mapping relationship between currently running PMs and VMs and improve resource utilization while reducing energy consumption and guaranteeing QoS. This paper takes the optimum mapping relation between VMs and PMs as the optimization objective, through minimizing energy consumption of PMs with the lowest host overloading risk; and resolves the optimization model with the proposed discrete DE algorithm. A Discrete DE algorithm-based VM placement scheme and method are presented. Through full mutation and crossover of the improved discrete DE algorithm, the optimum mapping relationship between VMs and PMs is fundamentally guaranteed, and the global optimization result is obtained. In addition, the above presented sub-algorithms, including host overload detection, VM selection, and under-loaded host detection algorithms, are integrated, and a hybrid, energy-efficient and quality-aware heuristic VM consolidation (EQ-VMC) method is proposed.

The detailed EQ-VMC method is as follows.

Algorithm 5: EQ-VMC
1: Input: H, H_{active}
2: Output: $Map < host, vm >$
3: $H_{over} \leftarrow MHOD(H_{active})$
4: $V_{mig} \leftarrow QVMS(H_{over})$
5: $H_{app} = H_{active} - H_{over}$
6: $Map < host, vm > \leftarrow Discrete\ DEVMP(V_j, H_{app})$
7: $H_s = H_{active} - H_{over} - H_{mig}$, $H_{sp} = H_{active} - H_{over}$
8: $Map_s < host, vm > \leftarrow ULHD(H_s, H_{sp})$
9: $Map < host, vm > \leftarrow Map < host, vm > \mathrm{U} Map_s < host, vm >$

In fact, the four sub-algorithms (Discrete DEVMP, MHOD, QVMS, ULHD) integrate and co-operate to yield the EQ-VMC method. Although there is an incomplete double-loop body in sub-algorithms MHOD, QVMS, and ULHD, theoretically speaking, the time consumption of the EQ-VMC method mainly comes from the Discrete DEVMP algorithm. Hence, the time complexity of EQ-VMC is identical to that of the well-known DE algorithm; i.e., $O(\delta * \sigma * \tau)$, in which δ is the number of iterations, σ is the population size, and τ is the dimension of the individual in the Discrete DEVMP algorithm. In practice, σ is the search space

Table 5
The properties of Bitbrains trace.

Date	Number of VMs	CPU		Memory		Bandwidth	
		Mean (%)	St.dev. (%)	Mean (%)	St.dev. (%)	Mean (%)	St.dev. (%)
2013-08-02	1237	7.20	5.97	8.83	4.35	0.76	1.84
2013-08-04	1233	8.05	4.83	9.75	4.12	0.80	1.77
2013-08-08	1209	10.27	6.64	9.69	4.52	0.70	1.67
2013-08-10	1205	8.17	5.43	9.47	4.36	0.85	2.00
2013-08-12	1202	9.57	6.36	9.25	3.86	0.79	1.79
2013-08-14	1194	4.88	4.54	8.41	3.33	0.59	1.41
2013-08-18	1189	8.39	3.73	8.95	3.16	0.89	1.88
2013-08-20	1186	8.99	3.45	9.10	3.05	0.79	1.46
2013-08-23	1176	9.44	4.64	9.74	3.90	1.01	2.19
2013-08-25	1175	5.48	4.06	8.40	3.63	1.12	2.39

Table 6
The properties of GoogleClusterTrace.

No.	Number of VMs	CPU		Memory	
		Mean (%)	St.dev. (%)	Mean (%)	St.dev. (%)
1	1200	1.63	0.49	2.16	0.10
2	1200	2.23	0.71	2.30	0.16
3	1200	2.71	0.82	2.31	0.16
4	1200	2.97	0.89	2.25	0.16
5	1200	3.01	0.91	2.27	0.17
6	1200	3.02	0.91	2.24	0.16
7	1200	2.99	0.91	2.23	0.17
8	1200	2.98	0.90	2.21	0.17
9	1200	2.96	0.91	2.20	0.16
10	1200	2.95	0.90	2.20	0.17

of the evolutionary algorithm—i.e., all of the probable mappings between VMs and PMs during ongoing VM consolidation; τ is the dimensionality of vector d_i^T—i.e., the number of PMs in data centers. Additionally, since the Discrete DEVMP algorithm is pulled by heuristic information, its convergence speed should be faster than that of the typical DE algorithm.

5. Experiment

5.1. Simulation setting

We utilized the CloudSim toolkit [36] as the simulation platform. In experiments, because the lower boundary on the number of VMs in the employed workload traces (e.g.,Bitbrains trace) is approximately 800, 800 heterogeneous PMs are created in cloud data centers, these PMs include four types, and the specific configuration and the power measurement are shown in Table 2, Table 3 (SPEC) [37]respectively. Besides, six types of VMs are created based on the instance parameters provided by EC2 (Amazon EC2 Product Details) [38], Table 4 shows their configuration parameters. These VMs are randomly deployed on different PMs.

In order to properly simulate the real resource request and response in data centers, three different data sets (Bitbrains trace [39], GoogleClusterTrace [40] and Alibaba cluster data [41]) are employed to examine the experiments. Bitbrains includes CPU usage, memory, and bandwidth workload; its selected 10-day trace is shown in Table 5. GoogleClusterTrace only includes CPU and memory usage. We evenly select 10-day data traces, and 1200 items of data trace are selected randomly from each day. The selected trace is shown in Table 6. Alibaba cluster data also includes CPU and memory usage, and we use the same operation to extract samples as that done on GoogleClusterTrace. The selected trace is shown in Table 7. Additionally, the parameters in EQ-VMC are set as follows, $F = 1$, $CR = 0.5$, $\alpha = 6$, $\beta = 0.5$, $\rho = 0.5$, $\omega = 0.8$, $total\_times = 10$, $S = 10 \times |H|$ and $EXP = 0.7$.

5.2. Evaluation metrics

We use five performance indices to evaluate performance in our experiments: SLA violation time per active host (SLATAH), performance degradation due to migration (PDM), SLA violations (SLAV), energy consumption (EC), and energy and SLA violations (ESV) [3,9].

(a) SLATAH, given by (21), measures the service quality of a running PM.

$$SLATAH = \frac{1}{n}\sum_{j=1}^{n}\frac{l_j^{violation}}{l_j} \tag{21}$$

where $l_j^{violation}$ is the SLAV duration period resulting from overloaded CPU resources for a physical machine h_j, l_j is the running time of the physical machine h_j, and n is the number of PMs.

(b) PDM, which indicates the extent of VM migration-related performance decline, is defined as

$$PDM = \frac{1}{m}\sum_{i=1}^{m}\frac{R_i^{mig}}{R_i} \tag{22}$$

where R_i^{mig} represents the size of unsatisfied demand for CPU resources as a result of the migration of a given virtual machine v_i, R_i is the size of total demand for CPU resources from v_i, and m is the number of VMs.

(c) SLAV measures the QoS of a data center on a single day.

$$SLAV = SLATAH \times PDM \tag{23}$$

SLATAH, PDM, and SLAV are inversely proportional to QoS.

(d) The comprehensive evaluation index ESV is formulized as (24), which reflects the energy consumption, VMMs, and service quality.

$$ESV = EC \times SLAV \tag{24}$$

where EC denotes the energy consumption of a data center in a single day, which is determined according to formula (4) in Section 3.2. A low ESV value indicates that more energy is saved and guarantees QoS of data centers.

Since VMs always suspend their services during live VM migration, prolonged VM migrations can further impact QoS. Degrading the number of insignificant VM migrations facilitates improving QoS. Therefore, if limited VM migrations can yield ideal effects given a VM consolidation method, this means that the corresponding VM consolidation method is highly efficient.

5.3. Experiment results

In this section, several experiments are listed, in order to validate the effectiveness and efficiency of the proposed optimization model and VM consolidation method from different vantage points.

Table 7
The properties of Alibaba cluster data.

No.	Number of VMs	CPU		Memory	
		Mean (%)	St.dev. (%)	Mean (%)	St.dev. (%)
20180606–0	1200	1.39	0.40	5.69	0.55
20180606–1	1200	1.38	0.38	5.46	0.56
20180606–2	1200	1.33	0.37	5.34	0.54
20180726–0	1200	2.46	0.71	9.28	0.31
20180726–1	1200	2.34	0.68	8.97	0.30
20180726–2	1200	2.38	0.72	9.30	0.30
20180815–0	1200	2.51	0.74	9.38	0.32
20180815–1	1200	2.32	0.69	8.95	0.32
20180815–2	1200	2.63	0.75	9.32	0.34

Table 8
Simulation results using Bitbrains trace with four metrics.

	EC (kWh)	SLATAH	VMMs	ESV
EQ-VMC	37.31	1.35	1842.2	0.00889
EC-VMC	37.33	1.64	6068.3	0.01197
ACS-VMC	39.39	2.62	3502.9	0.04915
SA-VMC	40.75	1.70	10603.3	0.02241
COFFGA	52.70	1.60	10827.9	0.02363

Table 9
Simulation results using Alibaba cluster data with four metrics.

	EC (kWh)	SLATAH	VMMs	ESV
EQ-VMC	38.83	0.43	2074.9	0.00086
EC-VMC	45.19	1.05	19810.6	0.02714
ACS-VMC	49.35	1.31	6409.1	0.01898
SA-VMC	233.55	0.23	13112.7	0.00215
COFFGA	60.19	0.57	8421.0	0.00208

5.3.1. Validating the optimization model

To validate the effectiveness and efficiency of the proposed optimization model and VM placement method, this section performs comparison between the presented Discrete DEVMP, PABFD, and FFD. We combine and evaluate each of them with 4 identical host overload detection methods and 3 VM selection methods, resulting in a total of 36 different combinations. The host overload detection algorithms [2,3] include benchmarks IQR, LR, MAD, and ST; the VM selection algorithms [2,3] include MC, MMT, and RS. These combined schemes are separated into 12 groups, each group with 3 different color columns corresponds to Discrete DEVMP, PABFD and FFD, respectively. The experimental comparisons using 36 different combination schemes are performed with the same measurements (EC, SLATAH, PDM, SLAV, ESV and VMMs) on the Bitbrains trace. The results are shown in Figs. 2–7.

Figs. 2–7 are the comparison results on various performance indices. Fig. 2 shows the results based on the EC index between the compared schemes. As shown in Fig. 2, the EC indices of schemes which use Discrete DEVMP are the lowest. Fig. 3 compares the SLA violation of all 36 combinations. The Discrete DEVMP algorithm performs best in all 36 combinations. For each method, Fig. 4 shows VMMs during VM consolidation. The times of VM migrations with Discrete DEVMP are the smallest. Fig. 5 shows that the PDM-driven performances of combination schemes which use Discrete DEVMP are the best, and the schemes using PABFD have the worst performance. Fig. 6 shows a comparison of SLAV on the different schemes. Since the SLAV depends on both SLATAH and PDM, the Discrete DEVMP algorithm outperforms other schemes in terms of SLATAH, PDM, and SLAV indices. Fig. 7 shows the ESV index, and the Discrete DEVMP algorithm again performs best.

The figures above show that the presented method is superior according to various indices; this validates its effectiveness of the proposed optimization model and method. In contrast, the VM placement policy employed by PABFD does not sufficiently predict the host overloading risk, resulting in frequent VM migration and a large number of SLA violations. In addition, due to the local greedy selection of PABFD, it is unable to reduce the number of running PMs immediately, resulting in energy waste. Additionally, as shown in the figures, several combination schemes using LR show little variation in EC, SLAV, and ESV indices, especially in SLATAH. The most unexpected changes come from the combination of LR and FFD. The primary reasons for this are that the LR algorithm predicts the resource utilization of PMs via linear regression. When the FFD algorithm performs VM placement according to the prediction results, although it greatly increases the resource utilization, it also leads to the occurrence of excessive VM consolidation and causes deterioration of QoS.

5.3.2. Effectiveness and efficiency evaluation

5.3.2.1. Comparisons in different evaluation metrics. In this section, we compare the proposed EQ-VMC method with four VM consolidation methods, i.e., ACS-VMC [6], SA-VMC [10], EC-VMC [9], COFFGA [7]. The comparison is examined using different data trace on the same indices respectively.

Table 8 shows the experimental results on the Bitbrains trace with the four indices. For the EC metric, the EQ-VMC method is the best, the EC-VMC method is the second best, and the COFFGA method is the worst. EQ-VMC and EC-VMC perform very similarly in EC, because they all take the total EC of data centers as one of the optimization objectives for VM consolidation. The COFFGA method aims at minimizing the number of running PMs and resource waste, which does not directly optimize energy consumption, and thus the solution search easily gets trapped in local optima. As for the SLATAH index, EQ-VMC performs the best, and ACS-VMC is 94% more than EQ-VMC, though its EC is only 5.6% more than EQ-VMC. This is because ACS-VMC aims to decrease the number of running PMs, which can significantly improve resource utilization. As to the VMMs index in Table 8, EQ-VMC has the smallest number , and ACS-VMC has the second smallest. In terms of the estimation of host overloading risk, EQ-VMC is capable of keeping the workload stable for running PMs, which further reduces VM migrations. ACS-VMC establishes an optimization model which contains the optimization objective of minimizing VM migrations, which causes it to relieve host overloading risk, even though it has less VM migrations. SA-VMC and COFFGA do not consider the QoS directly when it optimizes VMs placement, so they have a mediocre performance on VM migrations. ESV is a combination metric covering both energy and QoS for VM consolidation. EQ-VMC still has the best performance in ESV; EC-VMC has the second best. The EQ-VMC and EC-VMC methods are superior to other compared methods in both EC and QoS indices, because reducing energy consumption and relieving host overloading risk are both optimization objectives of their established optimization model. This proves that both EQ-VMC and EC-VMC methods realize their optimization objective.

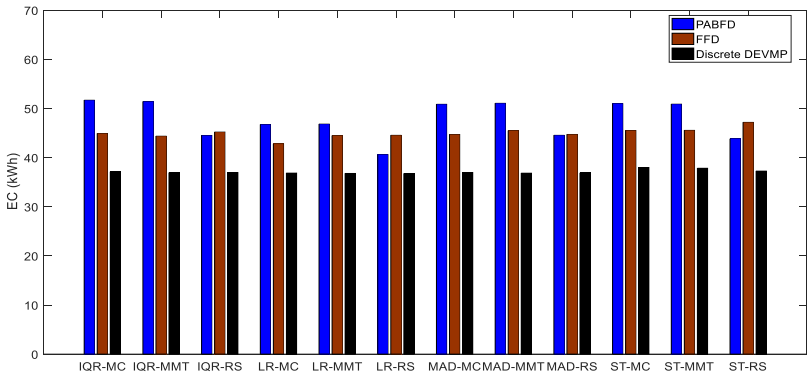


Fig. 2. Comparison of EC . (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

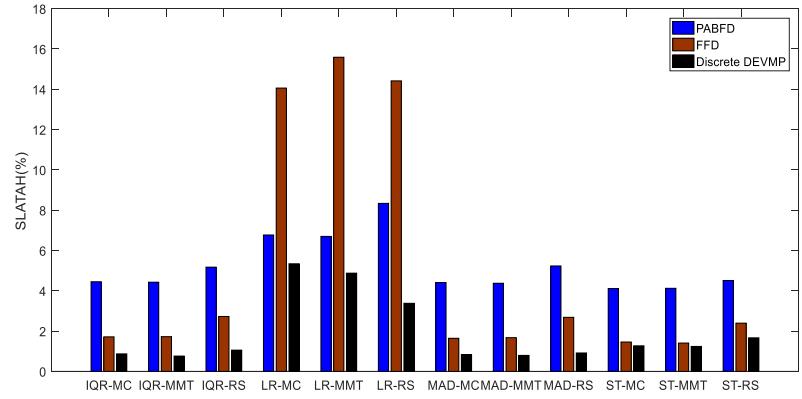


Fig. 3. Comparison of SLATAH. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

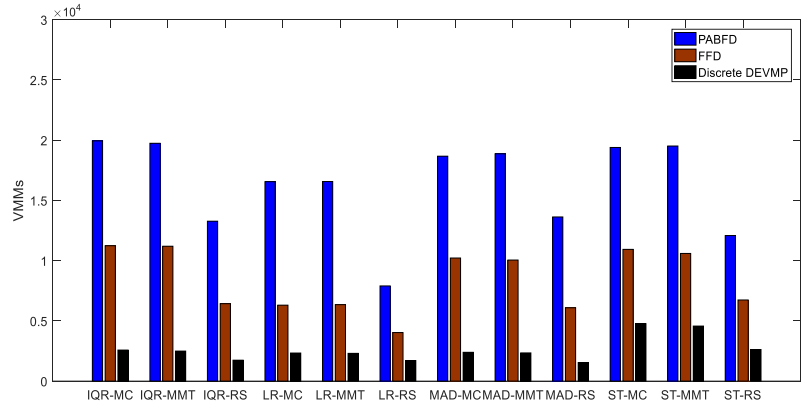


Fig. 4. Comparison of times of VM migrations. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

In summary, EQ-VMC shows better performance and stability in EC, SLATAH, VMMs, and ESV than other methods.

Table 9 shows the experimental results on Alibaba cluster data. In terms of the EC, VMMs, and ESV indices, EQ-VMC is superior to the other compared algorithms. Due to the constraint of overloading probability in EQ-VMC, overloading risk occurrences in running PMs are reduced. Notably, insignificant VM migrations decrease drastically, resulting in higher energy efficiency and QoS. As for the SLATAH indices, EQ-VMC is inferior to SA-VMC. The optimization objective of SA-VMC is to obtain the optimum balance for the consumption of various resources of running PMs. In contrast, Alibaba cluster data is a trace with a high memory-occupying ratio, in which memory utilization is three to six times higher than CPU utilization. Namely, as for Alibaba cluster data, it is difficult for SA-VMC to finally realize the optimization objectives, so it cannot effectively decrease the number of PMs, which results in increasing the energy consumption of data centers. Therefore, the low index of SLATAH for SA-VMC is precisely due to the data center still having too many running PMs after VM consolidation, resulting in an effective reduction of the overloading risk.

Table 10 shows the experimental results on GoogleClusterTrace. On the EC and SLATAH indices, EQ-VMC is inferior to EC-VMC. In terms of the VMMs and ESV indices, EC-VMC and ACS-VMC are superior to EQ-VMC. We note that on the GoogleClusterTrace, all of the indices by EQ-VMC are inferior to that of EC-VMC, but on Bitbrains trace EQ-VMC is superior to that of EC-VMC. The primary reasons are as follows: First, in GoogleClusterTrace, the

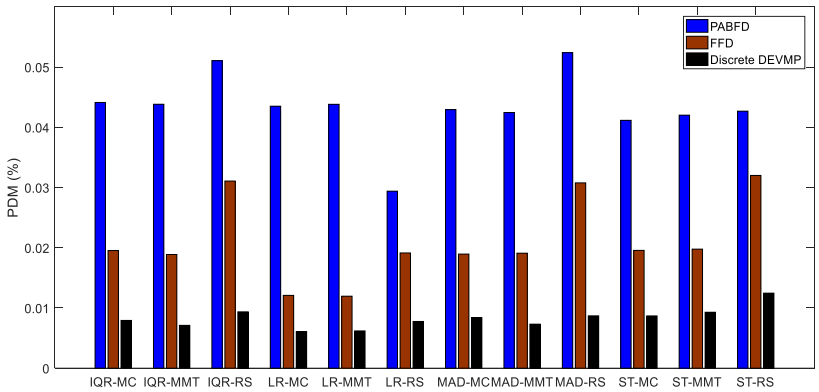


Fig. 5. Comparison of PDM. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

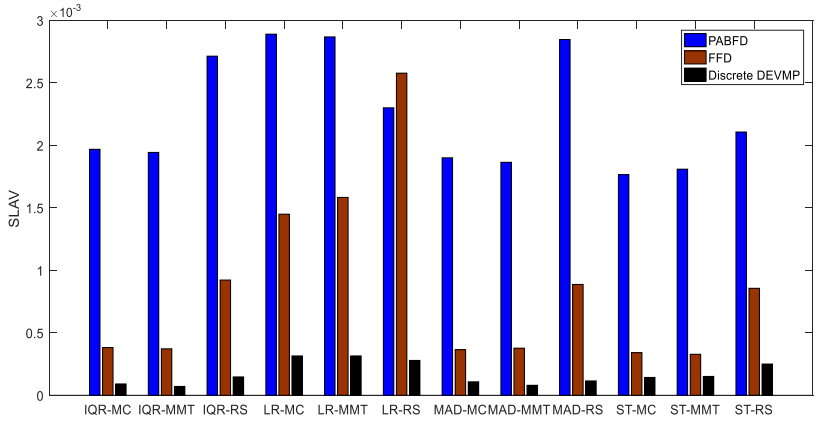


Fig. 6. Comparison of SLAV. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

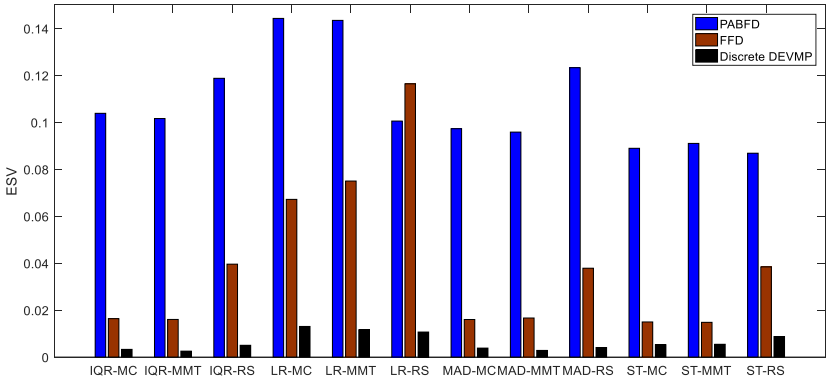


Fig. 7. Comparison of ESV. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 10
Simulation results using GoogleClusterTrace with four metrics.

	EC (kWh)	SLATAH	VMMs	ESV
EQ-VMC	22.31	1.34	5298.4	0.00446
EC-VMC	22.03	0.63	2865.7	0.00120
ACS-VMC	24.25	1.70	2395.1	0.00334
SA-VMC	27.17	1.84	14030.2	0.00997
COFFGA	35.76	1.91	14952	0.02155

requested CPU resource is lower and more stable, and the average CPU utilization is nearly 3%. This situation is not likely to cause host overloading or even host overloading risk. Thus, the host overloading probability constraint is meaningless. Next, EC-VMC takes host overloading probability as a constrain condition to PMs rather than taking host overloading probability as an optimization objective like that in EQ-VMC. This encourages the running PMs to improve resource utilization while avoiding host overloading risk. Finally, because of the unique workload characteristics, such as the Gaussian distribution in GoogleClusterTrace [33], the resource usage of running PMs does not have strong randomness, and even ACS-VMC has better results in VMMs and ESV indices than EQ-VMC.

Fig. 8 compares different schemes in terms of PDM that depend on VMMs and the memory capacity of the migrated VMs. Fig. 8(a) shows the PDM metric on the Bitbrains trace. The PDM of EQ-VMC is the lowest and most stable among compared methods,

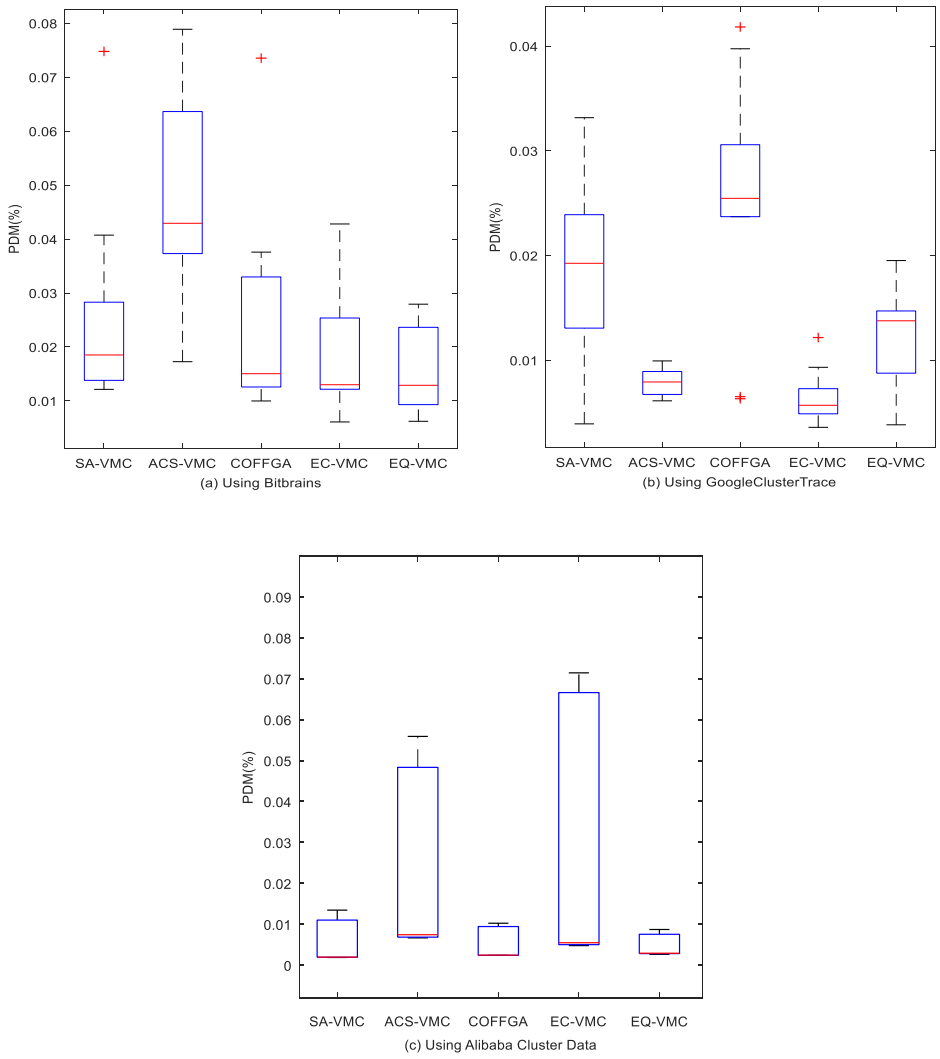


Fig. 8. Comparison of PDM.

and the performance of EC-VMC is the second lowest. The number of VMMs using ACS-VMC shown in Table 8 is also lower than that using EC-VMC, SA-VMC, and COFFGA, but PDM of ACS-VMC is the highest among all compared methods. The reason is that ACS-VMC has to migrate VMs with higher resource demand to reduce VM migration. SA-VMC and COFFGA use MMT for VM selection, so they obtain a lower PDM, despite the increase in VM migrations during VM consolidation. Fig. 8(b) shows the PDM results on GoogleClusterTrace. As we can see, EC-VMC is the best, and EQ-VMC locates the medium level. The primary reason for this is that, when the host overloading probability of PMs is small, the proposed optimization model in EQ-VMC is unable to choose proper destination PMs for VM placement. Unfortunately, this further spurs host overloading risk, frequent VM migrations, and consequently a higher PDM value. Finally, compared to SA-VMC and COFFGA, EQ-VMC still performs better. Fig. 8(c) shows the PDM metric on Alibaba cluster data. The PDM of EQ-VMC is the lowest and most stable among compared methods, and COFFGA performance is the second lowest. The VMMs of EC-VMC shown in Table 9 is higher than that of EQ-VMC, SA-VMC, and COFFGA, and its PDM is the highest among all compared methods. This is because the EC-VMC frequently performing VM migration due to the increase of memory utilization is much faster than the increase of CPU utilization during ongoing VM consolidation. Only in this way can host overloading risk be avoided.

Fig. 9(a), 9(b), and 9(c) show the comparison results on the SLAV index using Bitbrains trace, GoogleClusterTrace, and Alibaba cluster data respectively. We can see that, using Bitbrains trace and Alibaba cluster data, EQ-VMC is the best; this result indicates the optimization model proposed in this paper effectively facilitates relieving host overloading risk of running PMs. On GoogleClusterTrace, the SLAV value of EC-VMC is the best, and the SLAV of EQ-VMC is at a medium level among all compared methods. The main reason for this is the extreme workload characteristics in GoogleClusterTrace, namely, GoogleClusterTrace has a lower average CPU utilization, resulting in a very low probability of overloading risk occurrence. Resource overloading probability constraints are not sensitive to heuristic evolutionary optimization-like algorithms such as EQ-VMC.

In summary, by comparing SA-VMC, ACS-VMC, COFFGA, EC-VMC, and EQ-VMC on the same indices, we can see that EQ-VMC is capable of decreasing energy consumption and guaranteeing QoS for data traces which have high overloading risk, even if it is inferior on a few indices to other methods when performing on GoogleClusterTrace-like data traces which have low resource utilization.

5.3.2.2. Effectiveness evaluation. To evaluate the practical application of the proposed approach, this section further compares EQ-VMC and other methods on number of running PMs, number

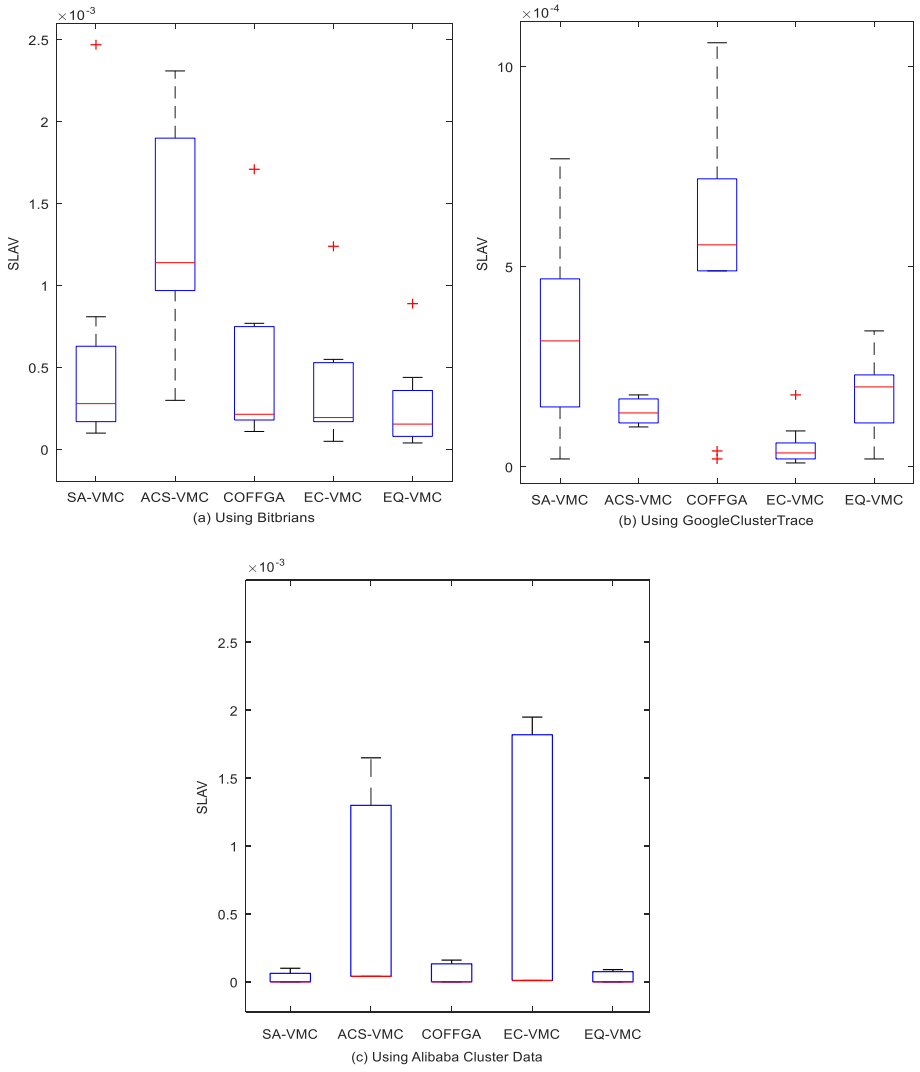


Fig. 9. Comparison of SLAV.

of VMMs, CPU resource utilization, memory resource utilization, and bandwidth utilization during ongoing VM consolidation. In our simulation, a cycle of VM consolidation is conducted every 5 min, and the total amount of cycles is 288 per day. Figs. 10–18 show the results.

Fig. 10 shows the variation in the number of running PMs during each cycle of ongoing VM consolidation using Bitbrains trace. Fig. 10(a) shows the variation from the 1st to 288th cycle of VM consolidation. With all five VM consolidation methods, the number of running PMs in data centers significantly decreases at the initial stage, thus reducing energy consumption. The variation from the 20th to 288th cycle is compared in detail in Fig. 10(b). As shown in the figure, in the initial stages, ACS-VMC, EC-VMC, and EQ-VMC have similar changing trends for the number of running PMs, but EQ-VMC has fewer running PMs than the other methods, especially in the later stages; this is also verified by the experimental results in Table 8. The EC index in Table 8 demonstrates that fewer running PMs results in lower energy consumption in data centers. In contrast, COFFGA and SA-VMC have greater numbers of running PMs than the other compared methods, because they employ FFD for VM placement that can easily be trapped in local optima, leading to high host overloading risk.

Fig. 11 shows the variation in the number of running PMs with respect to each cycle of ongoing VM consolidation, using Alibaba cluster data. Fig. 11(a) shows the variation from the 1st to the 288th cycle of VM consolidation. The number of running PMs in data centers significantly decreases at the initial stage for all five VM consolidation methods, among which EC-VMC decreases the most and the fastest, thus reducing energy consumption. The dramatic variation of running PMs in EC-VMC is because the EC-VMC algorithm aims to optimize energy efficiency and reduce virtual machine migration under the constraint of overload probability. In EC-VMC, the energy consumption model established depends on CPU utilization, which means that increasing the CPU utilization of a single PM facilitates the reduction of the total energy consumption in data centers. The Alibaba data tends to increase memory utilization, which causes the EC-VMC algorithm to constantly attempt to reduce the number of PMs and improve CPU utilization during VMs consolidation. However, excessive memory utilization continues to induce host overloading risk. So, the two aspects above create a contradiction. Consequently, the number of PMs in the data center fluctuates. The variation from the 20th to 288th cycle is further displayed in detail in Fig. 11(b), which shows that, besides the EC-VMC, the other compared algorithms have similar changing trends for the number of running PMs, with EQ-VMC having the least number of running PMs; this

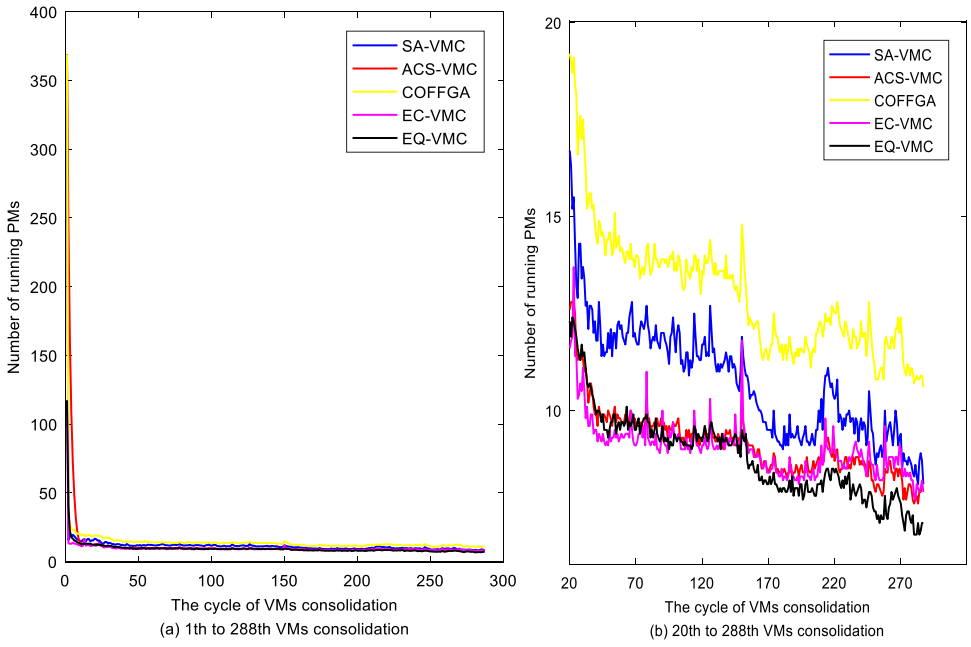


Fig. 10. The number of running PMs varying with the cycle of VM consolidation on Bitbrains.

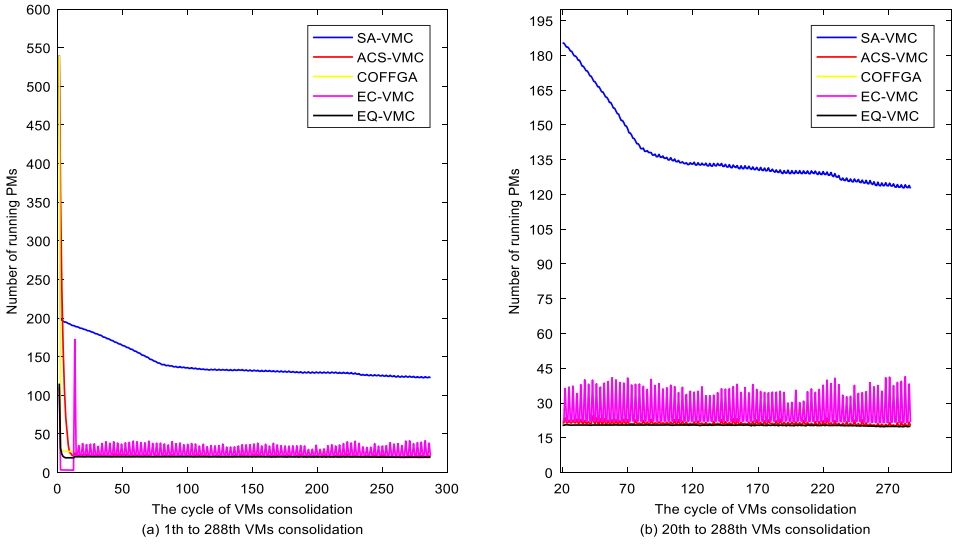


Fig. 11. The number of running PMs varying with the cycle of ongoing VM consolidation using Alibaba cluster data.

is also voted by the experimental results in Table 9. During the later cycles, a comb-like waveform can be clearly seen for EC-VMC. This is due to the same factors analyzed in Fig. 11(a). An additional factor is the embedded mechanism of EC-VMC, which is that it prefers to deploy migrated VMs from the under-loaded PMs onto destination PMs with a higher workload.

Fig. 12 shows the variation in the number of running PMs within each cycle of the ongoing VM consolidation on GoogleClusterTrace. As shown in Fig. 12(a), the number of running PMs in data centers also is significantly reduced in the initial stage. Fig. 12(b) further shows the variation from the 20th to the 288th cycle of ongoing VM consolidation. We can find that the number of running PMs of COFFGA and SA-VMC is apparently more than the other methods. This is due to the same reasons as mentioned before, namely, the regular distribution and low CPU utilization in GoogleClusterTrace. In contrast, the number of running PMs of EQ-VMC, EC-VMC, and ACS-VMC are similar, and less than that of the other compared methods. With the combined analysis of Table 10, the EC index of ACS-VMC is still larger than that of EQ-VMC and EC-VMC. This is because the employed optimization model in ACS-VMC is unable to determine the proper destination PMs for live VM migration. Unlike ACS-VMC, EC-VMC utilizes an ABC-like (e.g., Artificial Bee Colony) search method to find the optimum VM placement scheme with lowest EC and VMMs. In contrast, the EQ-VMC algorithm can effectively reduce the number of running PMs, both on the Bitbrains trace and GoogleClusterTrace, due to its embedded discrete DEVMP algorithm, which is able to find the optimum VM placement scheme with the lowest energy consumption and smallest host overloading risk.

On the one hand, the reduction of the number of running PMs during VM consolidation usually means that a single PM has to undertake more computing tasks, and its CPU resource utilization inevitably increases. Fig. 13 shows the variation of CPU utilization

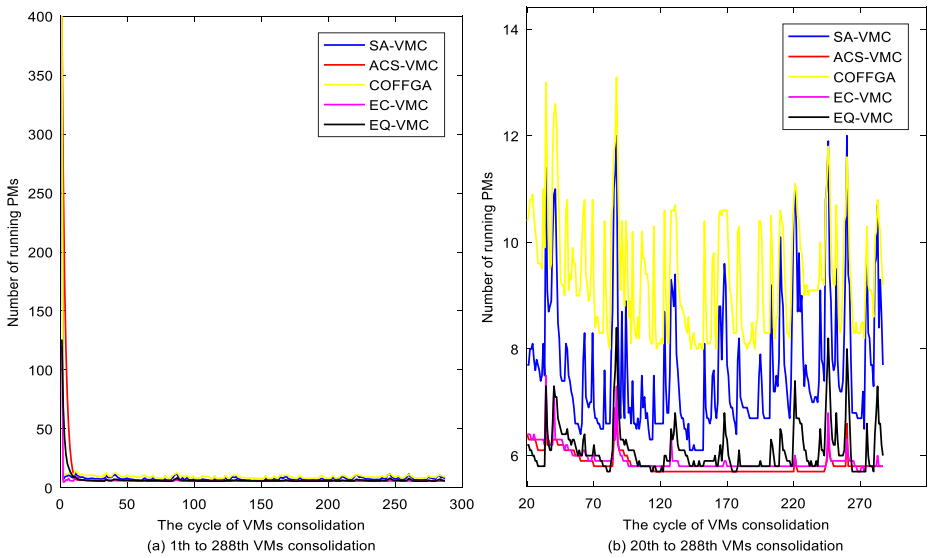


Fig. 12. The number of running PMs varying with the cycle of ongoing VM consolidation using GoogleClusterTrace.

on the running PMs during ongoing VM consolidation. On the other hand, the increasing of CPU resource utilization indicates higher energy consumption for that PM. Fig. 13(a) shows the results using Bitbrains trace; Fig. 13(b) shows the results with GoogleClusterTrace, and Fig. 13(c) shows the results using Alibaba cluster data. Like ACS-VMC, EC-VMC methods also have an obvious difference with the other compared algorithms; most notably, the EC-VMC method has a distinguish change during the initial cycles of VM consolidation. The VM consolidation methods show a very similar variation trend in CPU utilization during the remaining cycles. However, as for the Alibaba cluster data, the comb-like waveform still occurs with the EC-VMC method. We think that this is because of the high ratio for memory occupation in the Alibaba cluster data, which can easily result in host overloading risk, and also because of the VM placement scheme of EC-VMC. EC-VMC prefers to deploy migrated VMs onto destination PMs with higher workload. In contrast, as shown in Fig. 13(a), the CPU utilization of ACS-VMC almost reaches its highest peak during all cycles of VM consolidation. This means that ACS-VMC is sure to experience more host overloading risk than that of the other compared methods. With the combined analysis of the number of running PMs in Fig. 12, it is obvious that the capacity of physical resources of running PMs selected by the ACS-VMC method is smaller than that of EC-VMC and EQ-VMC, resulting in a greater number of running PMs. ACS-VMC also shows the highest CPU utilization on GoogleClusterTrace in Fig. 13(b) due to the same aforementioned reason of its extreme distribution. The reasons that the CPU utilization of all algorithms on the Alibaba cluster data is much lower than the other two data traces are as follows: High memory resource utilization is one of the characteristics of Alibaba cluster data. Although the EC-VMC algorithm improves the CPU utilization of PMs at the initial stage, after the initial VM consolidation, the high memory resource utilization frequently induces host overloading risk, resulting in the number of PMs not to steadily decrease, but to fluctuate. In this way, although the overloading of memory resources is effectively controlled, the CPU utilization of PMs is low (see Table 7). Similarly, the CPU utilization of PMs for other compared algorithms is also low for these same reasons.

Fig. 14 demonstrates the memory utilization of the running PMs during the cycles of VM consolidation. Fig. 14(a) is the results with the Bitbrains trace, Fig. 14(b) is the results with GoogleClusterTrace, and Fig. 14(c) is the results with Alibaba cluster data. Since there is no bandwidth trace in GoogleClusterTrace and Alibaba cluster data, Fig. 15 only shows the experimental results on bandwidth utilization using Bitbrains trace. Combining Fig. 13(a), Fig. 14(a), and Fig. 15, we can see that the CPU and memory utilization of the EQ-VMC method is at a medium level. This performance results from the stable workload yielded by the EQ-VMC method, which is accompanied with efficient network transmission and communications. It is also demonstrated by the high bandwidth utilization in Fig. 15. In contrast, ACS-VMC shows the highest resource utilization in CPU and memory, which causes frequent VM migration, higher host overloading risk, and QoS degradation. Besides the high host overloading risk, frequent VM migration also consumes extra network bandwidth, which results in a higher bandwidth utilization for the ACS-VMC method, shown in Fig. 15. In Fig. 14(c), besides SA-VMC displaying a relatively slow upward trend, the remaining algorithms have a drastic change in memory utilization during the initial cycles of VM consolidation. These phenomena result from an exaggerated VM consolidation, and a large amount of VMs hosted by the running PMs. EQ-VMC displays relatively high memory utilization due to the effective constraint on the overloading probability. SA-VMC clearly shows the lowest memory utilization. With the combined analysis in Fig. 11, it can be found that there are still many PMs running after VM consolidation by the SA-VMC algorithm, so its memory resource utilization is also lower. This is mainly because the optimization objectives of SA-VMC algorithm are not fully realized.

Additionally, as for the EQ-VMC method using GoogleClusterTrace, we get the same indices distribution due to the same aforementioned reasons that yield the outputs in Fig. 13(b) and Fig. 14(b).

VMMs reflect the stability of QoS. Generally, the fewer the VMMs, the better the QoS is. Fig. 16 shows the variation of VM migrations during ongoing VM consolidation using Bitbrains trace. Fig. 16(a) describes the variation trend from the 1st to 288th cycle. For a clear comparison, Fig. 16(b) is a partial of Fig. 16(a), showing the variation trend from the 20th to 288th cycles of VM consolidation. As shown in Fig. 16(a), since many PMs have powered off in the initial stage, a massive number of VM migrations occur. This phenomenon illustrates that these compared methods perform well in the early stages of VM consolidation. After the 20th cycle of VM consolidation, the VMMs spurred by EQ-VMC

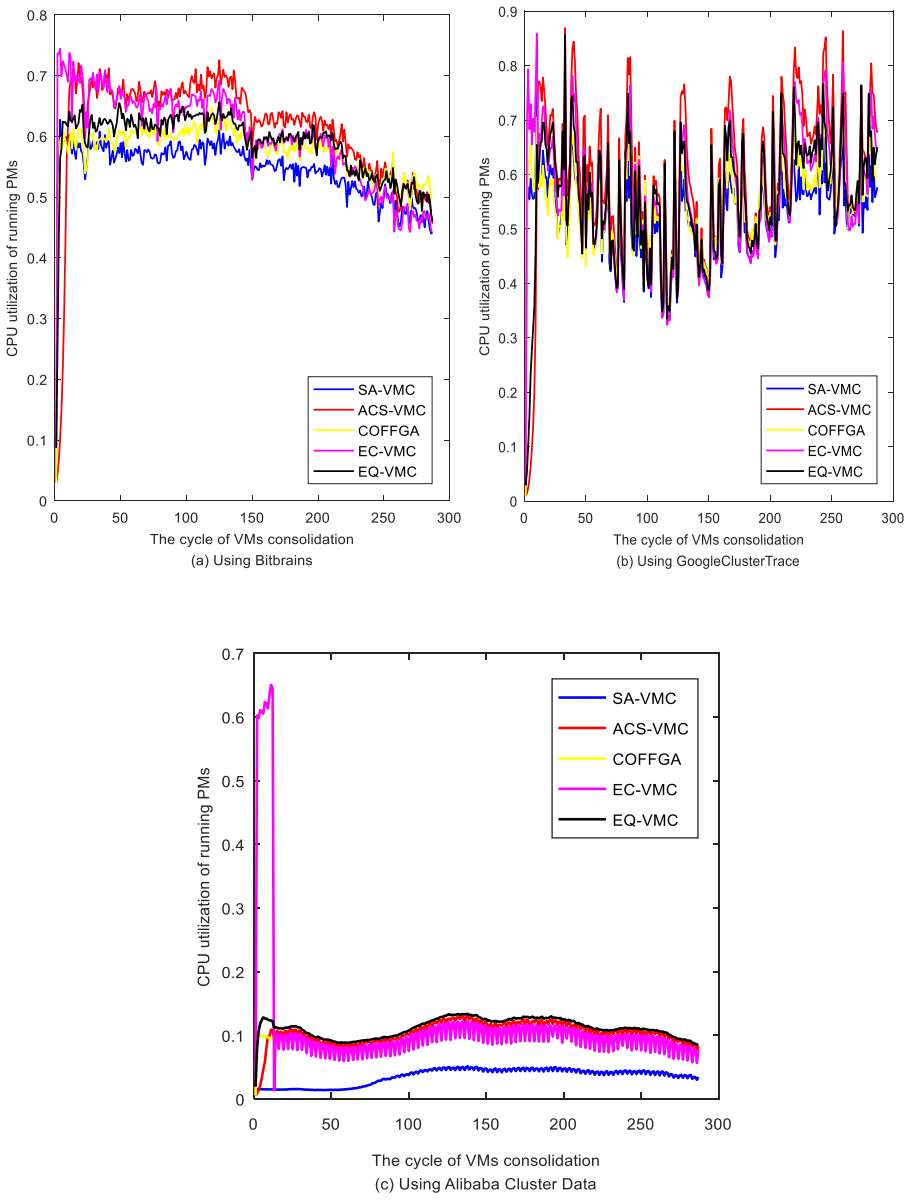


Fig. 13. CPU utilization of running PMs.

is lower than that of the other compared methods. This result indicates that the workload of PMs gets more stable after VM consolidation by EQ-VMC, so fewer virtual machines need to be migrated. In contrast, SA-VMC and COFFGA continue to conduct a large and inconsistent number of VM migrations; this can be demonstrated by their performance in QoS showed in Table 8, Figs. 8(a), and 9(a), of which is inferior to that of EQ-VMC.

Fig. 17 shows the variation of VM migrations during ongoing VM consolidations using GoogleClusterTrace. Within the figure, Fig. 17(a) shows the changes from the initialization to the 288th VM consolidation, and Fig. 17(b) is a partial of Fig. 17(a). ACS-VMC achieves the fewest VMMs, and its curve in Fig. 17 is smoother than other compared methods. This is mainly due to the lower resource request of CPU and the regular distribution of the workload in GoogleClusterTrace, which prevents ACS-VMC from suffering from host overloading risk, thus degrading the VMMs. The variation in VM migrations by EC-VMC throughout all 288 cycles of VM consolidations is simply inferior to that of the ACS-VMC method, but it does have a prominent advantage over ACS-VMC in EC and other QoS metrics, as shown in Table 10, Fig. 8(b), and Fig. 9(b). The curve of VM migrations with EQ-VMC throughout the 288 cycles of VM consolidation is mediocre among all the methods. However, EQ-VMC has significant improvement in EC compared to other methods.

Fig. 18 shows the variation of VM migrations during ongoing VM consolidations using Alibaba cluster data. Fig. 18(a) describes the variation trend from the 1st to 288th cycle. For a clear comparison, Fig. 18(b) is a partial of Fig. 18(a), showing the variation trend from the 20th to 288th cycles of VM consolidation. From Fig. 8, we can find three obvious phenomena: the dramatic change in the initial stage, the comb-like fluctuations curve for the EC-VMC method, and the extremely stable curve of the EQ-VMC method. These peculiar phenomena are explained in detail as follows. As shown in Fig. 18(a), since many PMs are powered off in the initial stage, a massive number of VM migrations occur. This illustrates that these compared methods perform well in the early stages of VM consolidation. As shown in Fig. 18(b), after the 20th cycle of VM consolidation, the number of VM migrations incurred by the EC-VMC method fluctuates noticeably between cycles. The primary reasons are analyzed as follows: The EC-VMC

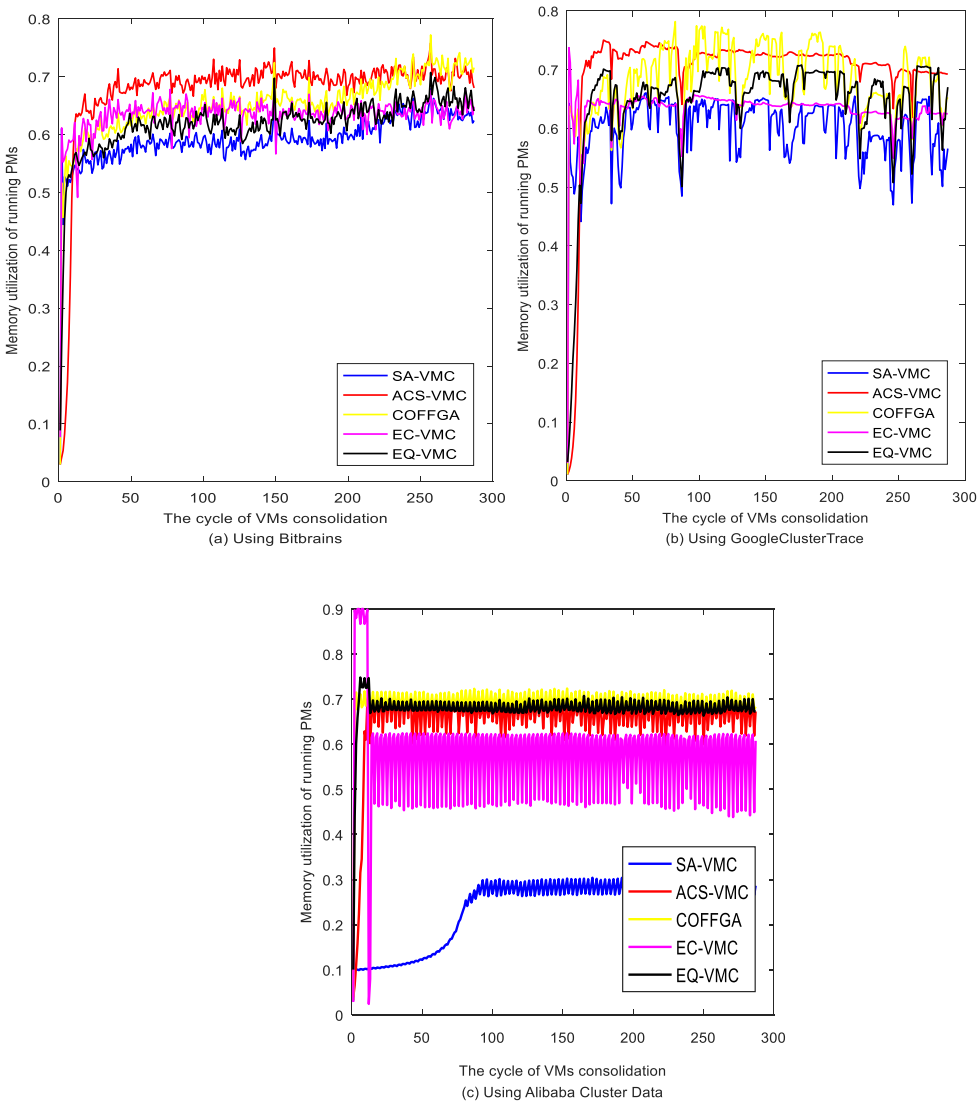


Fig. 14. Memory utilization of running PMs.

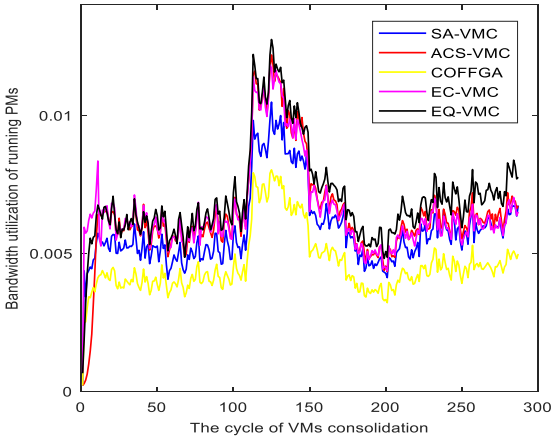


Fig. 15. Bandwidth utilization of running PMs.

algorithm mainly reduces the number of running PMs in data centers by increasing the CPU utilization of each single PM; this then reduces the total energy consumption. As for Alibaba cluster data, memory utilization increases much more quickly than CPU utilization. This situation results in the EC-VMC algorithm getting caught inside a bad loop; it first decreases the number of PMs to improve CPU resource utilization, and it then has to increase the number of PMs to meet the requested memory resource and relieve host overloading risk. Consequently, the number of VM migrations fluctuates in such a loop. The number of VM migrations spurred by the EQ-VMC method at each cycle of VMs consolidation is the minimum among all compared algorithms, maintaining less than about 20 migrations. In other words, the EQ-VMC method obtains a promising performance. One of the main reasons is that the probability of host overloading risk during ongoing VM consolidation is effectively kept under control by the optimization mechanism of EQ-VMC. Apart from this, due to the thorough search in the solution space by the proposed discrete DE algorithm, the mapping relationship between PMs and VMs is better optimized, thus guaranteeing a good QoS.

Based on the above analytical comparison, the results show that the proposed EQ-VMC method can effectively reduce energy consumption and guarantee QoS for data traces which have high overloading risk. The presented VM placement scheme efficiently maintains load balancing and relieves host overloading risk. Therefore, EQ-VMC realizes the optimization objectives.

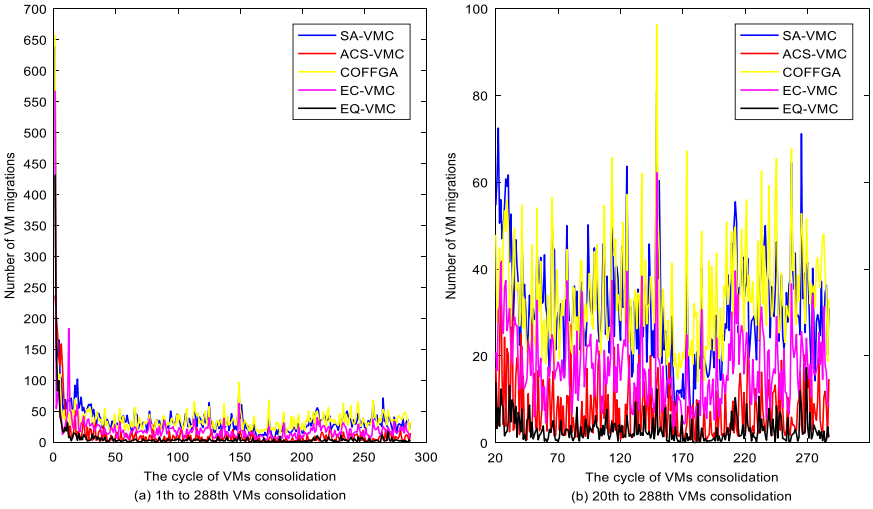


Fig. 16. The VMMs varying with the cycle of VM consolidation on Bitbrains.

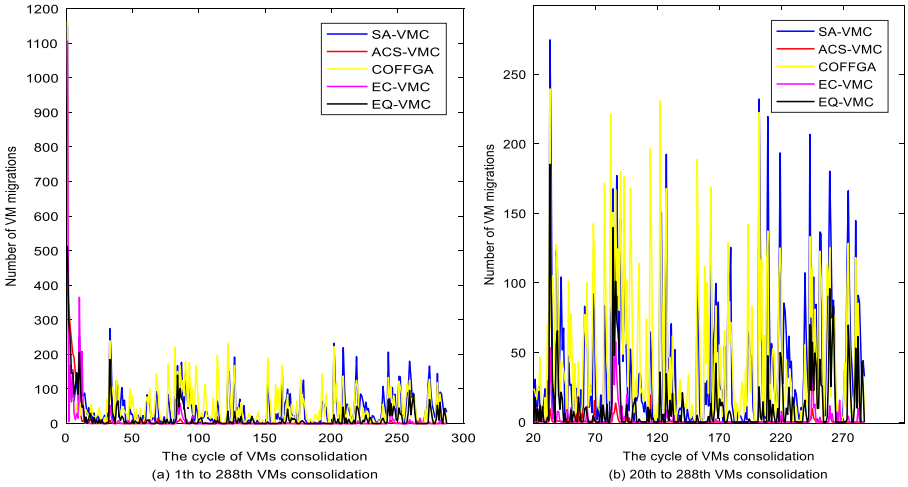


Fig. 17. The VMMS varying with the cycle of VM consolidation on GoogleClusterTrace.

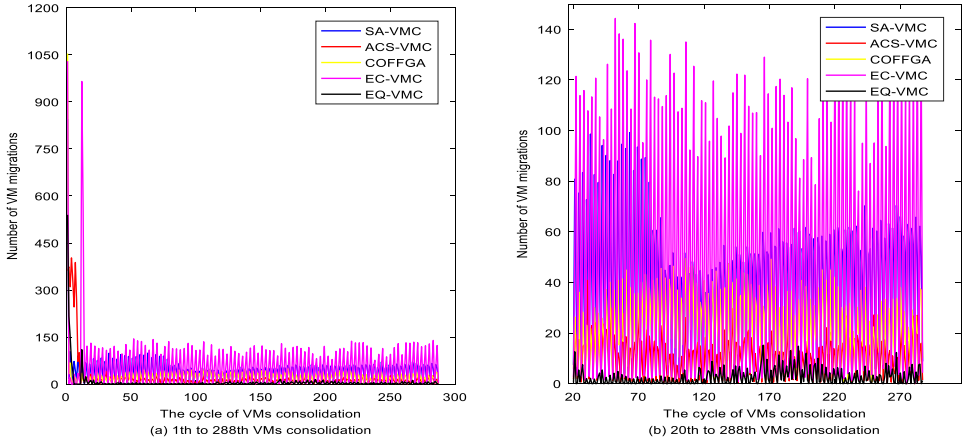


Fig. 18. The VMMS varying with the cycle of VM consolidation on Alibaba cluster data.

6. Conclusions and future works

Data centers provide access to shared resources on the Internet as a scalable, dynamic, and measurable service. VM consolidation performs live VM migration to appropriate destination PMs in order to improve one or more objectives and is thus regarded as an NP-hard problem. Generally, heuristic algorithms have shown superiority in resolving this complex combined optimization problem. Although VM consolidation using heuristic or

meta-heuristic algorithms which do not provide the best allocating solutions between VMs and PMs, they can offer near-optimal VM consolidation schemes.

This paper addresses VM consolidation with respect to heuristic evolutionary algorithms. First, our scheme aims to minimize the mathematical expectation of the energy consumption of running PMs while maintaining the lowest probable risk of host overloading, and to establish a dynamic optimization model for VM placement. Next, by abstracting the deployment relationship between each virtual machine and all physical machines into a single deployment vector, the deployment vectors of all VMs are thus equivalent to an item of mapping between VMs and PMs during a cycle of VM consolidation—namely, an individual in the heuristic evolution algorithm. Naturally, all probable mappings between VMs and PMs during ongoing VM consolidation correspond to a population, i.e., the search space of evolutionary computations. Afterwards, an improved discrete differential evolution (discrete-DE) algorithm is developed to resolve the above optimization model by finding the result in the search space which finds the optimum VM placement for the migrated VMs. Further, a VM placement algorithm is consequently proposed based on the presented optimization model. Finally, based on the above study, a hybrid heuristic evolutionary-based EQ-VMC method is developed for VM consolidation. Extensive trace-driven experiments are examined to validate the proposed method, and the experimental results demonstrate that it significantly reduces energy consumption, avoids unnecessary host overloading risk, and improves QoS for data traces which have high overloading risk.

However, there are several limitations that need to be further addressed in future works. First, this study only focuses on VM placement with the optimization objectives of minimizing energy consumption and relieving host overloading risk. Other optimization issues, such as minimizing VMMs and maximizing resource utilization during VM consolidation, need further study to improve the presented algorithm. Next, EQ-VMC shows prominent experiment results on the Bitbrains trace and Alibaba cluster data; but QoS is only at a mediocre level with GoogleClusterTrace. Thus the proposed optimization model should be given priority to adapt to GoogleCluster-like traces. Also, the GoogleClusterTrace should be deeply analyzed and data-mined for any extreme particularities, due to factors like the large scale of the Google data center.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work is supported by the Smart Manufacturing New Model Application Project Ministry of Industry and Information Technology (Grant No. ZH-XZ-180004), Future Research Projects Funds for the Science and Technology Department of Jiangsu Province (Grant No. BY2013015-23), the Fundamental Research Funds for the Ministry of Education (Grant No. JUSRP211A 41) and the 111 Project (Grant No. B2018)

References

[1] X. Zhu, D. Young, B.J. Watson, Z. Wang, J. Rolia, S. Singhal, 1000 islands: Integrated capacity and workload management for the next generation data center, in: International Conference on Autonomic Computing, IEEE Computer Society, 2008, pp. 172–181.

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

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

[4] H. Lin, X. Qi, S. Yang, S. Midkiff, Workload-Driven VM Consolidation in Cloud Data Centers, Parallel and Distributed Processing Symposium, IEEE, 2015, pp. 207–216.

[5] Z. Li, C. Yan, X. Yu, N. Yu, Bayesian network-based virtual machines consolidation method, Future Gener. Comput. Syst. 69 (2017) 75–87.

[6] F. Farahnakian, A. Ashraf, T. Pahikkala, P. Liljeberg, J. Plosila, I. Porres, Using ant colony system to consolidate vms for green cloud computing, IEEE Trans. Serv. Comput. 8 (2) (2015) 187–198.

[7] H. Hallawi, J. Mehnen, H. He, Multi-capacity combinatorial ordering ga in application to cloud resources allocation and efficient virtual machines consolidation, Future Gener. Comput. Syst. 69 (2016) 1–10.

[8] J. Jiang, Y. Feng, J. Zhao, K. Li, Dataabc: a fast abc based energy-efficient live vm consolidation policy with data-intensive energy evaluation model, Future Gener. Comput. Syst. 74 (2016) 132–141.

[9] Z. Li, C. Yan, L. Yu, X. Yu, Energy-aware and multi-resource overload probability constraint-based virtual machine dynamic consolidation method, Future Gener. Comput. Syst. 80 (2018) 139–156.

[10] S. Telenyk, E. Zharikov, O. Rolik, Consolidation of virtual machines using simulated annealing algorithm, in: International Scientific and Technical Conference on Computer Sciences and Information Technologies, IEEE, 2017, pp. 117–121.

[11] R.W. Ahmad, A. Gani, S.H.A. Hamid, et al., A survey on virtual machine migration and server consolidation frameworks for cloud data centers, J. Netw. Comput. Appl. 52 (C) (2015) 11–25.

[12] F. Farahnakian, P. Liljeberg, J. Plosila, LiRCUP: Linear regression based CPU usage prediction algorithm for live migration of virtual machines in data centers, in: Software Engineering and Advanced Applications, IEEE, 2013, pp. 357–364.

[13] X. Meng, C. Isci, J. Kephart, L. Zhang, E. Bouillet, D. Pendarakis, Efficient resource provisioning in compute clouds via VM multiplexing, in: International Conference on Autonomic Computing, Icac 2010, Reston, Va, Usa. DBLP, 2010, pp. 11–20.

[14] J. Cao, Y. Wu, M. Li, Energy efficient allocation of virtual machines in cloud computing environments based on demand forecast, J. Chin. Comput. Syst. 7296 (4) (2013) 137–151.

[15] A. Beloglazov, R. Buyya, Managing overloaded hosts for dynamic consolidation of virtual machines in cloud data centers under quality of service constraints, IEEE Trans. Parallel Distrib. Syst. 24 (7) (2013) 1366–1379.

[16] Y. Gao, H. Guan, Z. Qi, Y. Hou, L. Liu, A multi-objective ant colony system algorithm for virtual machine placement in cloud computing, J. Comput. System Sci. 79 (8) (2013) 1230–1242.

[17] M.R. Chowdhury, M.R. Mahmud, R.M. Rahman, Implementation and performance analysis of various VM placement strategies in cloudsim, J. Cloud Comput. 4 (1) (2015) 1–21.

[18] S.S. Masoumzadeh, H. Hlavacs, A Cooperative Multi Agent Learning Approach to Manage Physical Host Nodes for Dynamic Consolidation of Virtual Machines, Network Cloud Computing and Applications, IEEE, 2015, pp. 43–50.

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

[20] F. Farahnakian, T. Pahikkala, P. Liljeberg, J. Plosila, Energy aware consolidation algorithm based on k-nearest neighbor regression for cloud data centers, in: International Conference on Utility and Cloud Computing, IEEE, 2014, pp. 256–259.

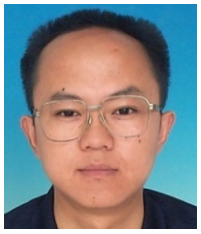
[21] S.S. Masoumzadeh, H. Hlavacs, An intelligent and adaptive threshold-based schema for energy and performance efficient dynamic VM consolidation, in: Energy Efficiency in Large Scale Distributed Systems, Springer Berlin Heidelberg, 2013, pp. 85–97.

[22] S.B. Shaw, A.K. Singh, Use of proactive and reactive hotspot detection technique to reduce the number of virtual machine migration and energy consumption in cloud data center, Comput. Electr. Eng. 47 (2015) 241–254.

[23] H. Shen, L. Chen, Distributed autonomous virtual resource management in datacenters using finite-Markov decision process, in: Proceedings of the ACM Symposium on Cloud Computing, ACM, 2014, pp. 1–13.

[24] S. Sohrabi, I. Moser, The effects of hotspot detection and virtual machine migration policies on energy consumption and service levels in the cloud, Procedia Comput. Sci. 51 (2015) 2794–2798.

[25] M. Li, J. Bi, Z. Li, Improving consolidation of virtual machine based on virtual switching overhead estimation, J. Netw. Comput. Appl. 59 (C) (2015) 158–167.
[26] M. Vitali, B. Pernici, U.M. O'Reilly, Learning a goal-oriented model for energy efficient adaptive applications in data centers, Inform. Sci. 319 (C) (2015) 152–170.
[27] M.A. Kaaouache, S. Bouamama, Solving bin packing problem with a hybrid genetic algorithm for VM placement in cloud, Procedia Comput. Sci. 60 (2015) 1061–1069.
[28] M. Mishra, A. Sahoo, On theory of VM placement: Anomalies in existing methodologies and their mitigation using a novel vector based approach, in: IEEE, International Conference on Cloud Computing, vol. 17, IEEE Computer Society, 2011, pp. 275–282.
[29] J.A. Aroca, A.F. Anta, M.A. Mosteiro, C. Thraves, L.Wang, Power-efficient assignment of virtual machines to physical machines, Future Gener. Comput. Syst. 54 (C) (2016) 82–94.
[30] F. Farahnakian, T. Pahikkala, P. Liljeberg, et al., Utilization prediction aware VM consolidation approach for Green cloud computing, in: 2015 IEEE 8th International Conference on Cloud Computing, IEEE, 2015, pp. 381–388.
[31] S.B. Melhem, A. Agarwal, N. Goel, M. Zaman, Markov prediction model for host load detection and vm placement in live migration, IEEE Access 6 (2018) 7190–7205.
[32] Z.A. Mann, Rigorous results on the effectiveness of some heuristics for the consolidation of virtual machines in a data center, Future Gener. Comput. Syst. 51 (2015) 1–6.
[33] L. Yu, L. Chen, Z. Z. Cai, et al., Stochastic load balancing for virtual resource management in datacenters, IEEE Trans. Cloud Comput. 99 (2016) 1.
[34] R. Storn, K. Price, Differential Evolution-a Simple and Efficient Adaptive Scheme for Global Optimization over Continuous Spaces, ICSI, Berkeley, 1995.
[35] K.Z. Ibrahim, Optimized pre-copy live migration for memory intensive applications, in: High PERFORMANCE Computing, Networking, Storage and Analysis, vol. 26, (b) IEEE, 2011, pp. 1–11.
[36] R.N. Calheiros, R. Ranjan, A. Beloglazov, C.A.F.D. Rose, R. Buyya, Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms, Softw. - Pract. Exp. 41 (1) (2011) 23–50.
[37] SPEC [Online]. https:// www.spec.org/power_ssj2008/results.
[38] Amazon EC2 Product Details [Online]. http://www.amazonaws.cn/en/ec2/details.
[39] S. Shen, V.V. Beek, A. Iosup, Statistical characterization of business-critical workloads hosted in cloud datacenters, in: 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, IEEE, 2015, pp. 465–474.
[40] [dataset] ClusterData2011_2 traces [Online]. Available: https://github.com/google/cluster-data/blob/master/ClusterData2011_2.md.
[41] https://github.com/alibaba/clusterdata/blob/master/cluster-trace-v2018/trace_2018.md.



Zhihua Li holds a Diploma and Ph.D. in Computer Science from Jiangnan University, Jiangsu, Wuxi, China in 2009. His research interests include network technology, parallel/ distributed computing, information security, data mining, pattern recognition. He is currently faculty of the Dept. of Computer Science and Technology at the Jiangnan University.

Xinrong Yu is a Ph.D. student in the Dept. of Computer Science and Technology at the Jiangnan University, Jiangsu, Wuxi, China.His research interests include cloud computing, parallel computing, distributed computing.

Lei Yu received his BS degree and MS degree in computer science from Harbin Institute of Technology, China. He is currently working towards the PhD degree in the School of Computer Science at Georgia Institute of Technology, USA. His research interests include cloud computing, data privacy, and sensor networks, wireless networks, and network security.

Shujie Guo is a M.S. student in the Dept. Of Computer Science and Technology at the Jiangnan University, Jiangsu, Wuxi, China. His current research interests include cloud computing, distributed computing.

Prof. Victor Chang is a Full Professor of Data Science and Information Systems, School of Computing, Engineering and Digital Technologies, Teesside University, Middlesbrough, UK, since September 2019. Previously he was a Senior Associate Professor (Reader), Director of Ph.D. (June 2016–May 2018), Director of MRes and Interim Director of BSc IMIS Programs at International Business School Suzhou (IBSS), Xi'an Jiaotong-Liverpool University (XJTLU), Suzhou, China. He joined XJTLU since June 2016. He is also a very active and contributing key member at Research Institute of Big Data Analytics (RIBDA), XJTLU and a key committee member at Research Center of Artificial Intelligence (RCAI), XJTLU. He is an Honorary Associate Professor at the University of Liverpool and Visiting Researcher at the University of Southampton, UK. Previously he worked as a Senior Lecturer at Leeds Beckett University, UK, for 3.5 years. Within 4 years, he completed Ph.D. (CS, Southampton) and PGCert (Higher Education, Fellow, Greenwich) while working for several projects at the same time. Before becoming an academic, he has achieved 97% on average in 27 IT certifications. He won a European Award on Cloud Migration in 2011, IEEE Outstanding Service Award in 2015, best papers in 2012 and 2015, the 2016 European award: Best Project in Research, 2016 SEID Excellent Scholar, Suzhou, China, Outstanding Young Scientist award in 2017, 2017 special award on Data Science, 2017 and 2018 INSTICC Service Awards and numerous awards since 2012. His research interests include IoT,Data Science,Cloud computing, Information security, Artificial Intelligence and Information system.