Cluster Computing (2024) 27:14269–14282
https://doi.org/10.1007/s10586-024-04670-6

An energy-aware ant colony optimization strategy for virtual machine placement in cloud computing

Lin-Tao Duan[1] · Jin Wang[1] · Hai-Ying Wang[1]

Received: 2 May 2024 / Revised: 20 June 2024 / Accepted: 6 July 2024 / Published online: 18 July 2024

Abstract
Virtual machine placement (VMP) directly impacts the energy consumption, resource utilization, and service quality of cloud data centers (CDCs), and it has become an active research topic in cloud computing. Inspired by the ant colony system (ACS) which has been proven effective metaheuristic approach for solving NP-hard problems, this paper proposes an improved ACS-based energy efficiency strategy (EEACS) for VMP problems. Our approach considers each virtual machine (VM) as an energy-consuming block, taking into account its individual energy requirements. EEACS ranks the physical machines (PMs) in a CDC in descending order based on their energy efficiency and optimizes both server selection and pheromone updating rules within the ACS. By guiding artificial ants towards promising solutions that balance energy consumption and resource utilization, EEACS ensures that VMs are placed efficiently based on pheromone and heuristic information. Extensive simulations in both homogeneous and heterogeneous computing environments demonstrate the effectiveness of our proposed strategy. The experimental results show that the EEACS enhances the resource utilization and achieves a notable reduction in energy consumption in comparison to conventional heuristic and evolutionary-based algorithms.

Keywords Energy efficiency · Resource utilization · Ant colony system · Virtual machine placement

1 Introduction

Cloud computing provides users with high performance computing, storage, and networking services through a flexible pay-as-you-go model [1]. It leverages virtualization technology to enable on-demand access to shared pools of configurable computing resources that can be rapidly provisioned and released with minimal management effort or service provider interaction. Currently, large organizations and companies are utilizing cloud data centers (CDCs) to offer a variety of cloud computing services, including infrastructure as a service (IaaS), platform as a service (PaaS) and software as a service (SaaS), to meet the diverse needs of their customers [2]. To meet the peak resource demands of customers, service providers (Amazon, Microsoft, Google, Baidu and Alibaba, etc.) continuously expand their CDCs by investing in additional hardware infrastructure, which leads to increased costs. However, resources demand exhibits fluctuating characteristics, resulting in a decline in the average resource utilization once the peak demand has subsided. Furthermore, the close arrangement of racks raises the temperature of processing units, requiring extra investments in optimizing cooling systems. As a result, this leads to further increases in CDC's energy consumption.

Virtual machines (VMs) run on physical machines (PMs) and act as separate, independent computing units, thus affording customers a flexible and scalable computing environment. Each VM is allocated a specific amount of the PM's resources, such as CPU, RAM, storage, and bandwidth, enabling it to execute its own operating system and applications independently. Cloud providers offer various VM sizes and configurations, allowing customers

✉ Lin-Tao Duan
duanlintao@cdu.edu.cn

Jin Wang
wangjin@cdu.edu.cn

Hai-Ying Wang
wanghaiying@cdu.edu.cn

1 School of Computer Science, Chengdu University, Chengdu 610106, People's Republic of China

to select the appropriate amount and type of resources for their specific workload requirements. The provision of VMs aims to provide infrastructure as a service, which enables customers to concentrate on their core business logic without having to manage infrastructure, while also benefiting from the elasticity, scalability, and cost-efficiency of the cloud. In CDC, the different solutions of virtual machine placement (VMP) have an impact on the energy consumption and resource utilization. Therefore, optimizing the VMP strategy is crucial for improving energy efficiency and resource utilization for CDC.

Commonly used VMP strategies in CDCs include First Fit Decreasing (FFD), Best Fit Decreasing (BFD), Least Loaded (LL), and Load Balancing [3]. The choice of VMP method depends on the specific requirements and objectives of the CDC. For instance, if resource utilization is a primary concern, FFD or BFD may be preferred. On the other hand, if load balancing and high availability are crucial, Load Balancing-based placement strategies may be more suitable. If energy efficiency is the optimization goal, numerous studies have provided effective metaheuristic approaches to solve the VMP problem, such as particle swarm optimization (PSO) [4], genetic algorithms (GA) [5], simulated annealing (SA) [6], discrete firefly algorithm (DFA) [7], ant colony optimization (ACO) [8, 9], and ant colony system (ACS) [10, 11], etc. These algorithms have been chosen due to their ability to handle complex optimization problems with multiple objectives.

In this paper, we propose an energy efficiency strategy based on ACS, called EEACS, for solving VMP multi-objective optimization problems after a comprehensive study of the above metaheuristic algorithms, as well as making contributions in the following mainly four aspects.

(1) We creatively treat each VM as an independent energy-consuming block, and the amount of energy requirement of each energy-consuming block is determined by both the duration of VM's existence and the server it is placed on.
(2) We devise a server energy consumption model based on energy-consuming block.
(3) To reduce energy consumption, we set pheromone and heuristic information using the reciprocal of the energy-consuming block's capacity, and dynamically deploy VMs based on ACS, enhancing the artificial ants' ability to explore new solution spaces and deeply leverage accumulated experience.
(4) To improve resource utilization, we first select an appropriate server from the pool of active servers based on ACS. If no suitable active server is found, we select the first powered-off server that has the highest energy efficiency and meets the VM's resource requirements, from a presorted pool arranged in descending order of energy efficiency. This approach effectively reduces the search space for servers, improving not only the deployment efficiency of VMs, but also minimizing the number of active servers.

The rest of the paper is organized as follows: In Sect. 2, we describe several works that are most related to our work. Section 3 provides a formal definition of VMP problem. Section 4 details the design of our energy efficiency algorithm EEACS. Then, in Sect. 5, we present our simulation results and evaluate our energy efficiency algorithm. Finally, this paper draws the conclusion and future work in Sect. 6.

2 Related work

VMP is a crucial issue for CDCs to process user requests using virtualization technology which enables multiple VMs to share the physical resources of a PM, thereby reducing the turnaround time of user tasks and improving the utilization of server resources. Currently, many studies formulate VMP as a multi-objective optimization problem [12–16]. Their goals includes improving resource utilization, reducing server energy consumption, minimizing the number of active servers, and enhancing the quality of service (QoS) that is defined by service level agreement (SLA).

However, there are N^M placement schemes for placing M VMs on N PMs. The algorithm's time complexity for finding the exact solution of the multi-objective optimization problem is $O(N^M)$. It is well-known that VMP problem is an NP-hard problem when N and M are relatively large [17]. Therefore, many literature adopt heuristic algorithms to find approximate optimal solutions in an acceptable time. Commonly used heuristic algorithms include greedy algorithms (e.g. FFD, LL) and evolutionary algorithms (e.g. PSO, SA, GA, DFA, ACS).

The FFD algorithm is a classic bin packing algorithm, which arranges items in descending order of size and then attempts to place each item into the first bin that can satisfy its requirements. This algorithm can reduce the number of bins required, thereby improving resource utilization. When applied to VMP, FFD arranges VMs in descending order of resource demand and places them on the first server that can meet their resource requirements. FFD can minimize the number of active servers required. Zhao et al. [18] combined statistical regression modeling, Pearson correlation coefficient analysis, and FFD methods to develop an algorithm called Segmented Iterative Correlation Combination for VM consolidation, which aims to improve resource utilization. The LL algorithm [3] is a load balancing algorithm that always assigns new tasks to the least loaded server such that each server can handle an appropriate amount of workload, avoiding situations where some servers are overloaded while others are idle. By applying the LL algorithm to VMP, performance degradation, increased latency, or service interruptions caused by server overload can be prevented. Azizi et al. [17] proposed a greedy randomized VM placement algorithm in large heterogeneous cloud environments. In order to optimize energy consumption and resource utilization, the algorithm sorts PMs based on their power consumption and traverses all PMs to calculate resource utilization after assigning VMs to them.

In comparison, evolutionary algorithms have better solutions in terms of energy efficiency when solving the VMP problem. Feng et al. [19] proposed a global energy-aware VMP strategy combining SA and GA for multi-objective optimization. They defined the system model and formalized the VMP problem as minimizing energy consumption for CDC. Their algorithm gradually optimizes VMP by setting control parameters, generating initial solutions, applying the Metropolis criterion, and executing the cooling process. Feller et al. [8] proposed an ant colony algorithm that uses an energy-aware strategy to optimize server energy consumption. Liu et al. [11] developed an ACS-based algorithm to guide artificial ants in finding optimized solutions that minimize the number of active servers. This algorithm combines local search techniques to enhance both search efficiency and solution quality. Xing et al. [9] proposed an ACO-based energy and traffic-aware algorithm that prioritizes VMs based on their traffic demands and selects PMs offering reduced power consumption and minimal network bandwidth usage. The algorithm takes into account both power consumption and network bandwidth as primary optimization objectives. Zhao et al. [20] developed a nonlinear energy consumption model based on server utilization and modeled the performance degradation of processors, memory, and network bandwidth under varying utilization. Finally, the authors propose a ACO-based VMP method to reduce server energy consumption and minimize performance degradation. To optimize energy consumption and avoid SLA violations, Zhou et al. [21] proposed an energy-aware VMP allocation and deployment mechanism for IOT applications and developed a four-threshold energy-aware framework that includes migrating VMs from lightly loaded servers and reallocating VMs from overloaded servers. To minimize the number of VM migrations and maximize server energy efficiency, Ajmera et al. [4] proposed a server residual efficiency-aware particle swarm optimization algorithm (SR-PSO). This algorithm comprises two distinct sub-algorithms: a VM selection algorithm and a server underload detection algorithm. The former is responsible for selecting VMs that have the potential to enhance server energy efficiency following migration, while the latter identifies underloaded servers exhibiting poor energy efficiency. The SR-PSO algorithm achieves optimal energy efficiency scheduling for VMs on servers by adjusting the classical PSO operators and incorporating a dual-objective fitness function during the exploration of the global solution space. Furthermore, the algorithm selectively reschedules only those VMs that may violate SLA or exhibit low server utilization within discrete scheduling intervals, thereby minimizing the risk of SLA violations.

Inspired by the literature [8, 10, 11], we propose an ACS-based energy efficient VMP algorithm EEACS aimed at minimizing server energy consumption and maximizing resource utilization. To achieve this, the following endeavors have been undertaken.

(1) In order to reduce server energy consumption, each VM is treated as an energy-consuming block. The reciprocal of the energy-consuming block's capacity is used as the initial value of pheromone and heuristic information on the path from the VM to the PM.
(2) In order to strengthen promising placement paths, EEACS employs a combined global and local pheromone update strategy. Additionally, to prevent rapid convergence to local optima, EEACS incorporates a rotation strategy, enhancing the ants' ability to explore alternative paths.
(3) In order to enhance resource utilization, EEACS incorporates two key strategies when searching for suitable servers for VMs. Firstly, EEACS prioritizes the selection of the most promising server from the pool of active servers. Secondly, in cases where no active server meets the VM's requirements, it selects the server with the highest energy efficiency from the pool of powered-off servers. This approach effectively narrows the search range for PMs, thereby improving the algorithm's performance.

3 VM placement

3.1 System model

In this paper, we assume that the CDC comprises N PMs. The ith PM in CDC, denoted as m_i, is described by a 5-tuple (pc_i, pm_i, P_i^{idle}, P_i^{max}, y_i), where pc_i represents the maximum number of CPU cores available on the server, pm_i indicates the RAM capacity of the server, P_i^{idle} and P_i^{max} denote the minimum and maximum power consumption of the ith PM during idle state and maximum utilization, respectively. Finally, y_i represents the current state of PM m_i, with a value of 0 indicating it is turned off and 1 indicating it is in an on state. VM v_j is represented by a 3-tuple (rc_j, rm_j, t_j), in which rc_j and rm_j represent the number of CPU cores and the size of RAM requested by the jth VM, respectively, and t_j is the maximum running time of the VM v_j. Each VM can be dynamically assigned to any PM in the CDC that is powered on and has sufficient remaining resources to meet the requirements of the VM. We use the notation $x_{i,j}$ to indicate whether v_j is placed on m_i. Specifically, $x_{i,j} = 1$ when v_j is placed on m_i, and $x_{i,j} = 0$ otherwise.

Finally, we use the formula below to characterize the state of a CDC that places M VMs on N PMs.

$$\begin{aligned} &CDC = \{PM, VM\}, \\ &PM = \{m_0, m_1, \ldots, m_i, \ldots m_N\}, \\ &VM = \{v_0, v_1, \ldots, v_j, \ldots v_M\}, \end{aligned} \tag{1}$$

subject to the following constraints:

$$\sum_{i=1}^{N} x_{i,j} = 1, \forall v_j \in VM, \tag{2}$$

$$\sum_{j=1}^{M} x_{i,j} \cdot y_i \geq 1,\ if\ y_i = 1, \forall m_i \in PM, \tag{3}$$

$$u_i^{cpu} = \frac{\sum_{j=1}^{M} rc_j \cdot x_{i,j}}{pc_i}, \tag{4}$$

$$u_i^{mem} = \frac{\sum_{j=1}^{M} rm_j \cdot x_{i,j}}{pm_i}. \tag{5}$$

Constraint (2) ensures that each VM can only be placed on one physical server at a time, and it also implies an assumption that any VM can find at least one server that meets its resource requirements. Constraint (3) indicates that a server in the on state must have at least one VM placed on it, implying that if no VMs are placed on a server, the server should be turned off to reduce energy consumption. Constraint (4) and (5) ensure that the CPU utilization u_i^{cpu} and memory utilization u_i^{mem} of PM m_i are less than or equal to 1, which implies that VMs cannot be assigned to PMs that cannot meet their resource requirements.

3.2 Power model

Multiple VMs share the system resources of a physical server to provide services for user requests within a specified time period. Therefore, the energy consumption of the server can be represented as the sum of the energy consumed during the execution of all VMs placed on it. As each VM is placed on a different PM, requests different system resources, and has different execution time, we can consider the VM as an energy-consuming block that consumes different amounts of electricity, where the size of the energy-consuming block is the amount of electricity required for its execution. Similar to the literature [3], we define the power consumption of server as a linear power consumption model based on the CPU utilization of the server, as shown below

$$P_i = P_i^{idle} + (P_i^{max} - P_i^{idle}) \cdot u_i^{cpu}. \tag{6}$$

Next, we assume server m_i has already deployed k VMs with k execution time $t_1 \leq t_2 \leq \cdots \leq t_k$. t_k represents the longest execution time among these VMs, and the execution time of the other VMs is less than or equal to t_k. Under these circumstances, the energy consumption of server m_i can be represented as

$$\begin{aligned} E_i = &\left(P_i^{idle} + (P_i^{max} - P_i^{idle})\frac{rc_1 + \cdots + rc_k}{pc_i}\right) \cdot t_1 \\ &+ (P_i^{idle} + (P_i^{max} - P_i^{idle})\frac{rc_2 + \cdots + rc_k}{pc_i}) \cdot (t_2 - t_1) \\ &+ \cdots + (P_i^{idle} + (P_i^{max} - P_i^{idle})\frac{rc_k}{pc_i}) \cdot (t_k - t_{k-1}). \end{aligned} \tag{7}$$

Then simplifying the formula, we obtain the following expression:

$$E_i = P_i^{idle} \cdot t_k + \sum_{j=1}^{k} (P_i^{max} - P_i^{idle})\frac{rc_j}{pc_i} \cdot t_j. \tag{8}$$

The energy consumption of a server is determined by the VMs placed on it. The total electricity consumed by all VMs running on the server represents the energy consumption of that server during that period of time. From Eq. (8), we obtain the contribution of VM v_j to the energy consumption of server m_i, as shown below.

$$\Delta E_{i,j} = \begin{cases} (P_i^{max} - P_i^{idle})\dfrac{rc_j}{pc_i}t_j,\ if\ t_j \leq t_k, \\ P_i^{idle}(t_j - t_k) + (P_i^{max} - P_i^{idle})\dfrac{rc_j}{pc_i}t_j, \\ if\ t_j > t_k, \end{cases} \tag{9}$$

where t_k represents the maximum execution time of all VMs residing on m_i, prior to the deployment of v_j. This contribution $\Delta E_{i,j}$ represents the amount of electricity required for v_j as an energy-consuming block, and also represents the increase in energy consumption of m_i when v_j is placed on it.

3.3 Problem definition

When tasks from clients and edge servers arrive at the CDC, the CDC creates VMs based on the resource requirements of these tasks (e.g., CPU cores, RAM, network bandwidth and storage, etc.) to serve these requests. Taking into consideration various factors such as server load, energy consumption, resource utilization, and QoS, the most suitable physical servers is selected to deploy these VMs. This paper defines the VMP problem as a multi-objective optimization problem that assigns M VMs to N PMs such that the total energy consumption is minimized and the resource utilization is maximized in the cloud computing environment for processing these tasks. To simplify the problem description and reflect the natural characteristics of time-varying resource requirements for tasks, we define the following three constraints:

(1) System resources only consider computation and storage resources. Tasks only request CPU cores and RAM resources, and server utilization only calculates the utilization efficiency of CPU and RAM.
(2) The resource requests $(rc_j,\ rm_j)$ of VM v_j will not exceed the resource capacity $(pc_i,\ pm_i)$ of PM m_i.
(3) The maximum execution time of tasks determines the running time t_j of the VM v_j.

The ultimate objective of our optimization algorithm is to minimize energy consumption and maximize resource utilization while ensuring that all current VMs are assigned to suitable PMs. Our objective function is shown as below.

$$\begin{cases} Minimize \sum_{i=1}^{N} (E_i \cdot y_i) \\ Maximize(u_i^{cpu} \cdot y_i,\ u_i^{mem} \cdot y_i). \end{cases} \tag{10}$$

We use an ACS-based optimization algorithm to solve VMP problem. The pheromone and heuristic information on the path from VMs to PMs induce the ants to allocate VMs to PMs according to the expected optimal path. By optimizing the representation rules of pheromone and heuristic information, our algorithm can achieve better results than other competing algorithms after a limited number of training iterations. Figure 1 illustrates the process of ants placing VMs onto PMs to construct an approximate optimal solution for VMP in our EEACS strategy.

In Fig. 1, the servers are depicted as rectangles, with their designated length and width reflecting the CPU and RAM capacities, respectively. Similarly, VMs are represented by cuboids, where the length, width, and height correspond to the requested CPU cores, RAM size, and execution time. When a VM is allocated to a server, the cuboid also represents an energy-consuming block, revealing the total energy consumed during its execution. In Fig. 1, we also find that the shaded rectangles in each server represent the remaining CPU and RAM of this server. We use the symbols ac_i and am_i to represent the remaining CPU and RAM of PM m_i, respectively, such that the available CPU and RAM of PM m_i can be obtained through the formulas $ac_i = pc_i - \sum_{j=1}^{M} (rc_j \cdot x_{i,j})$ and $am_i = pm_i - \sum_{j=0}^{M} (rm_j \cdot x_{i,j})$. Therefore, increasing the values of u_i^{cpu} and u_i^{mem} is equivalent to decreasing the values of ac_i and am_i. The process of an artificial ant searching for the current optimal server for a VM can be briefly described as follows: The artificial ant utilizes a strategy that balances exploitation and exploration, based on the pheromone concentration and heuristic value on the path from the VM to the active server, to find the most promising path to place the VM. If there is no active server that can satisfy the resource requirements of the VM, an energy-efficient server is selected from the pool of powered-off servers. Then, it is switched to the active state, and the VM is deployed on this server. This visualization depicted in Fig. 1, which represents PMs as rectangles and VMs as cuboids, effectively enhances comprehension of resource allocation and energy consumption patterns within the cloud computing environment.

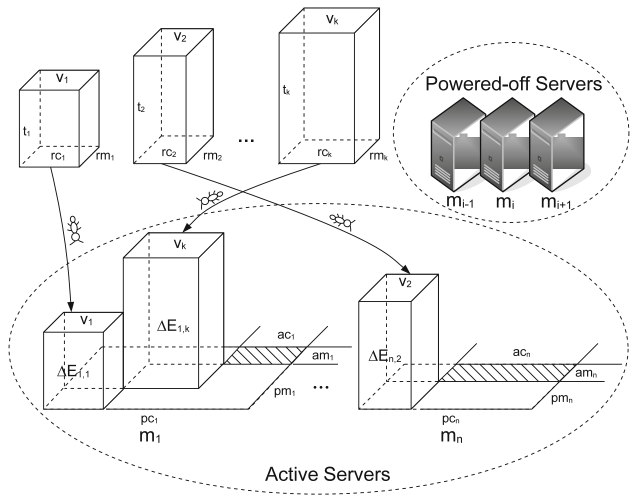

Fig. 1 Construction process of the solution to the VMP problem using EEACS strategy

4 Proposed algorithm

In this section, we present the heuristic value setting method, pheromone updating rule, and server selection rule. Finally, we present the complete algorithm pseudo code based on these methods and rules.

4.1 Heuristic information

The heuristic information is the control information that helps ants determine the most promising direction for VMP. Since one of our optimization goals is to minimize system energy consumption, we define heuristic information to allow ants to generate smaller energy-consuming blocks as much as possible when allocating VMs. Therefore, we use the inverse of the energy-consuming block's capacity as the calculation formula for heuristic information, which is show as follows:

$$\eta_{i,j} = \frac{1}{\Delta E_{i,j}}. \tag{11}$$

4.2 Pheromone updating rule

Pheromone records the historical preference information of each VM on each PM. If more ants choose to place v_j on m_i, the higher concentration of pheromone left on edge (v_j, m_i), the more likely it is to attract more ants to choose this placement path. Conversely, if fewer ants choose this path, the pheromone level on the path will decrease due to evaporation, making it less likely for more ants to choose it. Our optimization algorithm requires multiple iterations, with each iteration involving all ants completing the placement of all VMs. Our algorithm adopts both local pheromone updating rule and global pheromone updating rule during multiple iterations. Among them, a local pheromone update is performed after all ants complete the allocation of one VM, while a global pheromone update is performed after all ants complete the placement of all VMs (i.e., one iteration). Assuming R ants, M VMs, and I iterations are required, the optimization algorithm needs to complete $I \times R \times M$ local pheromone updates and I global pheromone updates.

After all ants complete the allocation of $v_j \in VM$, where $1 \leq j \leq M$, the local pheromone updating operation is performed on the allocation path from v_j to each server $\forall m_i \in PM$, where $1 \leq i \leq N$. The updating rule is as follows.

$$\tau_{i,j} = (1-\rho) \cdot \tau_{i,j} + \rho \cdot \tau_0 \cdot x_{i,j}, \tag{12}$$

where $0 < \rho < 1$ represents the pheromone decay parameter, while $\tau_0 = 1/\Delta E_{i,j}$ is the initial pheromone value on the path, which is equal to the inverse of the energy-consuming block's capacity. When an ant has assigned v_j to server m_i (i.e., $x_{i,j} = 1$), indicating that the path (v_j, m_i) has visited, the pheromone level on this path is increased by $\rho \cdot \tau_0$ after decay. Otherwise, if no ant deploys v_j to the server m_i (i.e., $x_{i,j} = 0$), only the decay operation is performed on the pheromone from v_j to the server m_i.

After completing one iteration, the global pheromone updating operation updates the pheromone on all paths from VMs to PMs. The global pheromone updating operation first finds the optimal solution S_{best} in this iteration, which is the solution that leads to the lowest system energy consumption among all ants' solutions, with its energy consumption denoted as E_{best}. Then, the pheromone on the placement paths of VMs to PMs covered by the optimal solution is reinforced, while the pheromone on the paths not in the optimal solution only undergoes evaporation according to the pheromone evaporation rate $0 \leq \alpha \leq 1$. This makes the optimal solution in this iteration more attractive during the next iteration. We normalize the optimal energy consumption as below:

$$NE_{best} = E_{best}/E_{max}, \tag{13}$$

where E_{max} is the system energy consumption in the worst-case scenario. The worst-case scenario refers to the situation where each PM only receives one VM, and the PM is the least energy-efficient machine ($\widehat{pc}$, $\widehat{P^{idle}}$, $\widehat{P^{max}}$, 1) in the CDC. Therefore, E_{max} can be calculated using the following formula.

$$E_{max} = \sum_{j=1}^{M} \left(\widehat{P^{idle}} + \left(\widehat{P^{max}} - \widehat{P^{idle}}\right) \cdot \frac{rc_j}{\widehat{pc}} \right) \cdot t_j. \tag{14}$$

Finally, we present the global pheromone updating rule as follows.

$$\tau_{i,j} = (1-\alpha) \cdot \tau_{i,j} + \alpha \cdot \frac{1}{NE_{best}} \cdot x_{i,j}. \tag{15}$$

4.3 Server selection rule

Our algorithm includes two optimization objectives: minimizing the overall energy consumption of the system and maximizing the utilization of server resources. By updating pheromones, ants are guided to place VMs on paths that can optimize energy efficiency. To achieve the goal of improving resource utilization, we rank the servers in descending order of energy efficiency, and use set A_j to represent all active servers that meet the resource requirements of placing v_j (i.e., $ac_i \geq rc_j$ and $am_i \geq rm_j$). If $A_j = \Phi$, we switch the first server in the powered-off state to the powered-on state and add it to A_j. Obtaining the set A_j can achieve the goal of placing all VMs on as few PMs as possible, effectively reducing the amount of remaining resources on each active server.

If there are more than one available servers (i.e., $|A_j| > 1$) that meet the requirements of placing v_j, we employ heuristic values and pheromones to assist ants in selecting more promising server within A_j for v_j. First, We define the probability of an ant selecting PM m_i for VM v_j as follows:

$$p_{i,j} = \frac{\tau_{i,j} \cdot \eta_{i,j}^{\beta}}{\sum_{m_u \in A_j} \tau_{u,j} \cdot \eta_{u,j}^{\beta}}, \ \forall m_i \in A_j. \tag{16}$$

In our EEACS algorithm, an ant places VM v_j on PM m_i from the available machines A_j according to the following state transition rule:

$$m_i = \begin{cases} \underset{m_u \in A_j}{argmax}\, \tau_{u,j} \cdot \eta_{u,j}^{\beta}, & if\ q \leq q_0 \\ A, & otherwise, \end{cases} \tag{17}$$

where q is a random number uniformly distributed in [0,1], A is a random server selected from A_j by a roulette wheel selection according to probability distribution given in (16) and q_0 is a parameter in [0,1] that determines the relative importance of exploitation versus exploration. When an ant decides which server to allocate a VM to, it generates a random number q using a uniformly distributed random number generator. If $q \leq q_0$, it chooses the server with the maximum product of pheromone and heuristic value determined by formula (17), which helps to further strengthen promising paths. If $q > q_0$, it chooses the server determined by formula (16), which helps the ant explore new path.

4.4 EEACS algorithm definition

The overall flowchart of the EEACS algorithm is presented in Fig. 2, which details the fundamental concepts of our method in five steps.

The first step is to initialize the CDC environment, which includes initializing environmental parameters, creating N PMs, and M VMs. All servers are initially in the

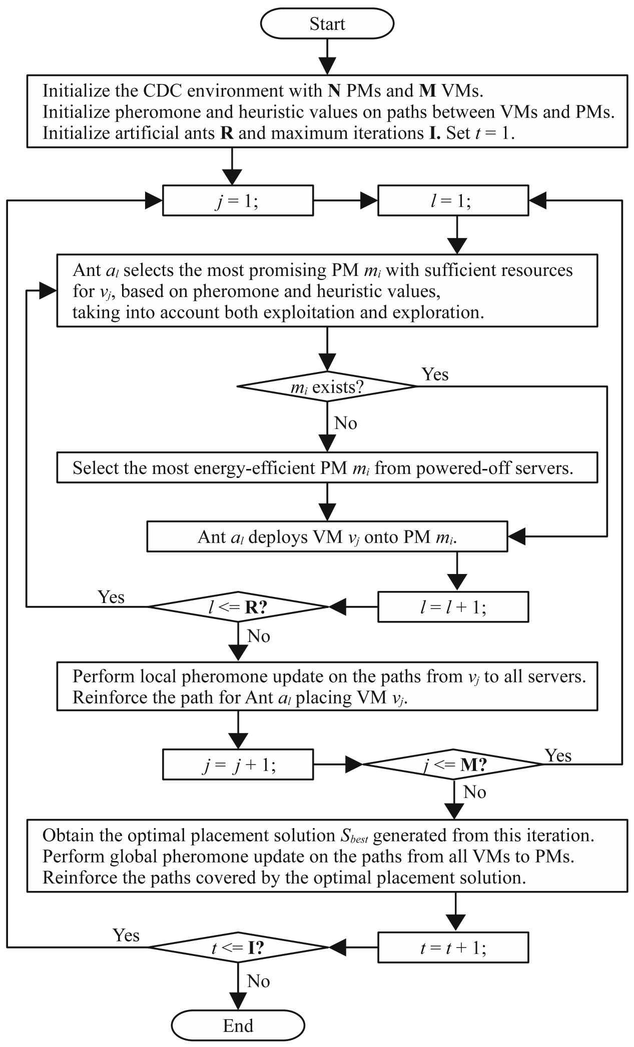
Start
Initialize the CDC environment with N PMs and M VMs.
Initialize pheromone and heuristic values on paths between VMs and PMs.
Initialize artificial ants R and maximum iterations I. Set t = 1.
j = 1;
l = 1;
Ant a_l selects the most promising PM m_i with sufficient resources for v_j, based on pheromone and heuristic values, taking into account both exploitation and exploration.
m_i exists?
Yes
No
Select the most energy-efficient PM m_i from powered-off servers.
Ant a_l deploys VM v_j onto PM m_i.
l <= R?
Yes
No
l = l + 1;
Perform local pheromone update on the paths from v_j to all servers.
Reinforce the path for Ant a_l placing VM v_j.
j = j + 1;
j <= M?
Yes
No
Obtain the optimal placement solution S_best generated from this iteration.
Perform global pheromone update on the paths from all VMs to PMs.
Reinforce the paths covered by the optimal placement solution.
t <= I?
Yes
No
t = t + 1;
End

Fig. 2 Flowchart of the EEACS algorithm

inactive state, and the number of active servers is zero. Therefore, the first server allocated to a VM is selected from the pool of powered-off servers. In the second step, each artificial ant searches for the most suitable server for each VM among all active servers, taking into account the pheromone concentration and heuristic value on the path from the VM to the PM. If no suitable server is found, the PM with the highest energy efficiency among the powered-off servers is chosen instead. The third step involves performing a local pheromone update after all ants have placed a VM. In the fourth step, once all ants have completed the placement of all VMs, the optimal placement scheme is selected from all the ant placement schemes as the optimal solution for the current iteration, followed by a global pheromone update to reinforce the path covered by the optimal solution, further influencing future ant placements. The fifth step serves as a decision point. If the maximum number of iterations has not been reached, the algorithm returns to the second step for the next iteration. However, if the maximum number of iterations has been met, the algorithm terminates.

To provide a more thorough understanding of the EEACS algorithm, Algorithm 1 presents the pseudo-code detailing its execution. The inputs to this algorithm are the set of VMs awaiting deployment and the set of PMs.

Firstly, a CDC is created encompassing the sets of PMs and VMs, followed by the initialization of parameters required for the algorithm, including I, R, y, A_j, q_0, α, β, ρ, etc. (line 1). Subsequently, the worst-case total system energy consumption E_{max} is calculated for placing these VMs on PMs (line 2).

The algorithm then proceeds to I iterations, where after each iteration, an optimal solution is obtained, and a global pheromone update is performed (line 4). Within each iteration, each VM is assigned to a server by R ants (line 5). To accomplish the allocation of a specific VM v_j, the algorithm enters R iterations, where in each iteration, an ant a_l is selected to finalize the placement of the VM v_j on the most desired server m_i from the available set A_j (line 6). To achieve this, the algorithm first selects an available server set A_j for each VM. A_j is initially empty, and the algorithm searches for all active servers in descending order of their energy efficiency (line 3). If the remaining resources of a server meet the requirements of v_j, it is added to A_j (lines 11–13). If $A_j = \Phi$ after searching through all active servers, the first powered-off server with the highest energy efficiency is selected and added to A_j (lines 16–18). If there are more than one available servers, a random number q is generated, and based on whether q is greater than q_0, the algorithm chooses between exploring a new solution or leveraging historical experience for faster convergence to select a server from A_j (lines 20–25). Subsequently, the VM v_j is placed on the chosen server m_i (line 27). After all ants have completed the deployment of a VM, a local pheromone update rule is performed (line 29). Once all VMs have been placed, each ant generates a solution to the VMP problem. Among these solutions, the one with the lowest total energy consumption is chosen as the optimal solution S_{best} for the current iteration, followed by a global pheromone update (lines 31–33). After I iterations, the final solution to the VMP problem and the total system energy consumption obtained by our algorithm are returned (line 35). Implicitly, this algorithm also assists in determining the final number of powered-on servers and the resource utilization efficiency of each server.

Algorithm 1 EEACS algorithm

```
Input: $PM, VM$
Output: global best solution $S_{best}$ and its $E_{best}$
1: initialize CDC and set $y = 0$ for each server;
2: compute $E_{max}$ through (14);
3: sort servers in $PM$ by power efficiency in decrease order;
4: for all $c \in (1...I)$ do
5:     for all $v_j \in VM$ do
6:         for all $a_l \in (1...R)$ do
7:             for all $m_i \in PM$ do
8:                 if $y_i = 0$ then
9:                     continue;
10:                end if
11:                if $ac_i \geq rc_j$ and $am_i \geq rm_j$ then
12:                    insert server $m_i$ into $A_j$;
13:                end if
14:            end for
15:            if $A_j = \Phi$ then
16:                power-on server $m_i$;
17:                insert server $m_i$ into $A_j$;
18:                directly select server $m_i$;
19:            else
20:                generate a random number q $\in$ [0,1];
21:                if $q \leq q_0$ then
22:                    select server $m_i \in A_j$ through (17);
23:                else
24:                    select server $m_i \in A_j$ through (16);
25:                end if
26:            end if
27:            ant $a_l$ places $v_j$ on $m_i$;
28:        end for
29:        update local pheromone through (12);
30:    end for
31:    obtain $S_{best}$ and $E_{best}$ among all ants' solutions;
32:    normalize $E_{best}$ to the interval [0,1] through (13);
33:    update global pheromone through (15);
34: end for
35: return global best solution $S_{best}$ and its $E_{best}$;
```

5 Experiments and evaluation

To evaluate our algorithm effectively, we simulated both homogeneous and heterogeneous computing environments. In the homogeneous environment, we utilized the DELL M820 server, which has 512 processing cores and 384 GB of RAM. This server consumes 832 W of power in an idle state, while its peak power consumption rises to 3997 W when fully utilized. For the heterogeneous environment, we employed a mix of DELL M820 and IBM NX360 servers. The IBM NX360 features 240 CPU cores and 288 GB of RAM, consuming 497 watts in idle mode and 2414 watts under full load. Table 1 presents detailed physical server specifications for our simulated homogeneous and heterogeneous cloud computing environments. In the two computing environments, we obtained the values of four

Table 1 Server details in the simulated cloud computing environments

Server	Cores	Memory (GB)	P^{idle} (W)	P^{max} (W)
DELL M820	512	384	832	3997
IBM NX360	240	288	497	2414

Table 2 Parameters of ACO and EEACS

Algorithm	R	q_0	α	β	ρ	τ_0
ACO	5	0	1	2	0.7	τ_{max}
EEACS	5	0.5	0.7	2	0.7	$1/\Delta E_{i,j}$

evaluation metrics of our algorithm EEACS in solving the problem of VMP, including the energy consumption of the CDC, the number of active servers, the utilization of CPU and RAM. We also compared these values with those of ACO, FFD and LL algorithms. The ACO algorithm adjusts pheromone levels and heuristic information based on the number of active servers and their available resources. The objective is to minimize the number of active servers, thereby reducing overall power consumption. The FFD algorithm places VMs on PMs based on their resource demands, while selecting PMs according to their available resources. To maximize resource utilization, VMs are placed on servers with the most appropriate size. On the other hand, the LL algorithm aims to balance the load across the system by selecting the server with the largest remaining resources each time. All algorithms have been implemented in Java and all experiments are conducted on a PM with an Intel Core i5 running at a frequency of 1.8 GHz and 4.0 GB RAM.

In order to ensure comparability, we standardized the parameter settings for all four algorithms in both homogeneous and heterogeneous environments. The simulation experiments deployed up to 600 VMs, with $N = M$ servers in each experiment. This guarantees that, in the worst-case scenario, each VM can be deployed on a separate server. Each VM was assigned a random number of processors between 1 and 128, RAM ranging from 0 to 100 GB, and a runtime between 30 and 60 s such that the ratio of total requirement of CPU and RAM of all VMs is close to 1:1. Both EEACS and ACO used five ants to search for the optimal path that is $R = 5$. The parameters for the EEACS algorithm were set to $q_0 = 0.5$, $\alpha = 0.7$, $\beta = 2$, $\rho = 0.7$, and $\tau_0 = 1/\Delta E_{i,j}$. The ACO algorithm utilized the parameter settings of $\alpha = 1$, $\beta = 2$, $\rho = 0.7$, and $g = 2$, which are consistent with the original literature [8]. The parameter settings of the ACO and EEACS algorithms in our experiments are shown in Table 2.

5.1 Homogenous environment

The DELL M820 server was used in a homogeneous computing environment. We varied the number of VMs from 100 to 600 and obtained placement solutions and evaluation metrics, including server energy consumption, number of active servers, CPU utilization, and RAM utilization, for the four algorithms. Figure 3 illustrates the total energy consumption of all active servers for the four algorithms. Compared to the greedy algorithms FFD and LL, the evolutionary algorithms EEACS and ACO achieved the lowest energy consumption. Although EEACS did not fully leverage the advantage of selecting servers based on energy efficiency in the homogeneous computing environment, it still demonstrated a significant advantage in saving energy, averaging 14.75%, 61.13%, and 60.93% less power consumption than ACO, FFD, and LL, respectively. It is important to note that EEACS consistently achieved lower server energy consumption than the other three algorithms across the range of VM counts from 100 to 600, indicating the stability of its energy-saving performance.

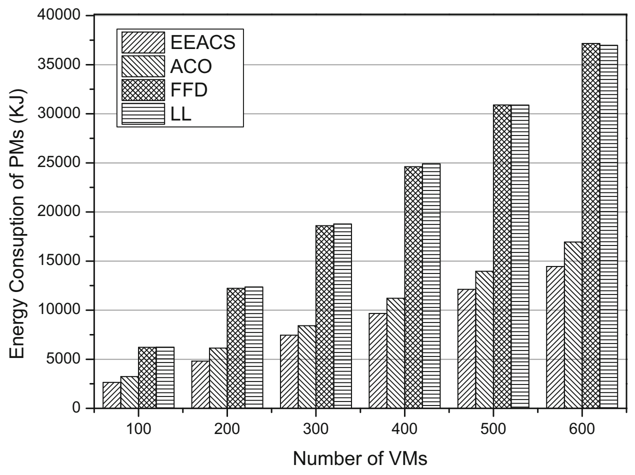

Fig. 3 Comparative results of total server energy consumption under four algorithms for six scenarios with VM sizes ranging from 100 to 600 in a homogeneous environment

Figure 4 compares the number of active servers for four algorithms across six VM sizes. As illustrated in Fig. 4, the FFD and LL algorithms utilize the fewest servers, while the performance of the EEACS algorithm is relatively close to that of the FFD and LL algorithms. When the number of VMs is 600, the EEACS algorithm requires three additional servers than the FFD algorithm. On average, across the six scenarios, the EEACS algorithm uses 2.17 and 2.00 more servers than the FFD and LL algorithms, respectively. The ACO algorithm performs poorly in this metric. When

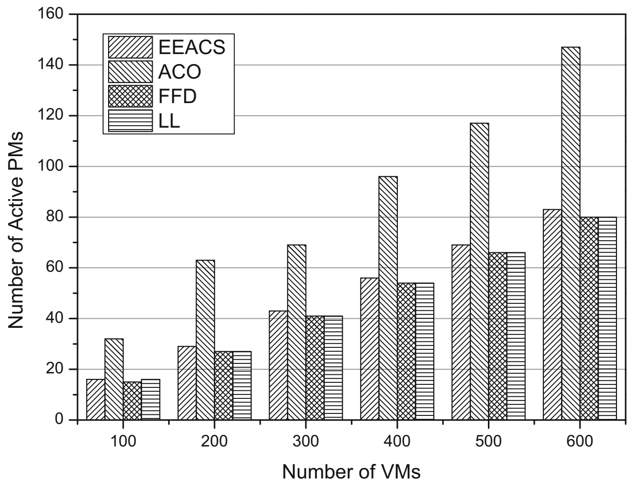

Fig. 4 Comparative results of the number of active servers needed to solve the VMP problems by four algorithms in a homogeneous environment

dealing with the same scale of VMP problems, the ACO algorithm employs the greatest number of servers. Under six different VM size scenarios, the average number of servers used by the EEACS algorithm is 43.51% less than that of the ACO algorithm.

Figure 5 compares the resource utilization of EEACS with the other three algorithms. As shown in the Fig. 5, across the six VM size scenarios, the average CPU and RAM utilization of EEACS are 89.64% and 89.67%, respectively, while those of ACO are 49.69% and 49.86%. FFD has the highest resource utilization, with 94.25% for CPU and 94.23% for RAM. EEACS achieves slightly higher average CPU and RAM utilization than the ACO algorithm. However, it is important to maintain a balance in resource utilization to avoid resource waste or excessive server load, which can lead to increased processor temperature and compromised system stability. The resource utilization achieved by our algorithms EEACS, as seen in Fig. 5, are within acceptable limits.

5.2 Heterogeneous environment

In a homogeneous environment, all servers exhibit identical performance. However, cloud computing environments typically consist of heterogeneous servers, which requires algorithms to not only minimize the number of active servers but also ensure that energy-efficient servers are utilized to provide services to users as much as possible. Consequently, further simulation experiments are being conducted in a heterogeneous environment comprising two distinct server types. The two types of servers are the DELL M820 and the IBM NX360, with IBM NX360 servers exhibiting greater energy efficiency than the DELL M820 servers. In order to ensure that both types of servers are utilized when solving the VMP problem with six different scales of VMs ranging from 100 to 600, the number of IBM servers was set to be one-tenth of the scale of VMs, with the remaining servers being DELL M820. In order to facilitate a meaningful comparison with the experimental results obtained in a homogeneous environment, the randomly generated parameters for VMs are aligned with the settings employed in the homogeneous environment. In particular, the CPU range of each VM is set to [1,128], the RAM range of each VM is set to [0,100], and the execution time range of each VM is set to [30,60].

Figure 6 illustrates the impact of four algorithms on the energy consumption of CDC for six scenarios with varying VM sizes ranging from 100 to 600. It is evident from the Fig. 6 that the FFD and LL algorithms yield suboptimal

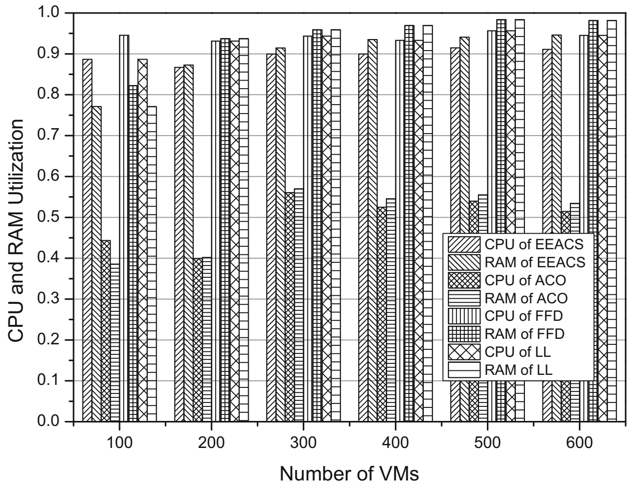

Fig. 5 Comparative results of CPU and RAM utilization of all servers due to four algorithms for solving the VMP problem in a homogeneous environment

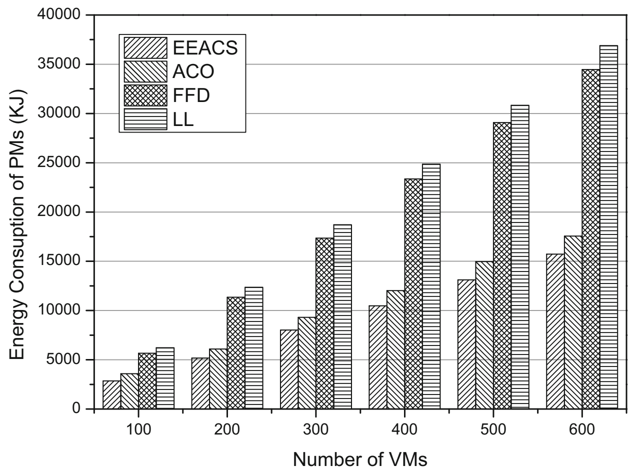

Fig. 6 Comparative results of total server energy consumption for VMP among four algorithms across six scenarios with VM sizes from 100 to 600 in a heterogeneous environment

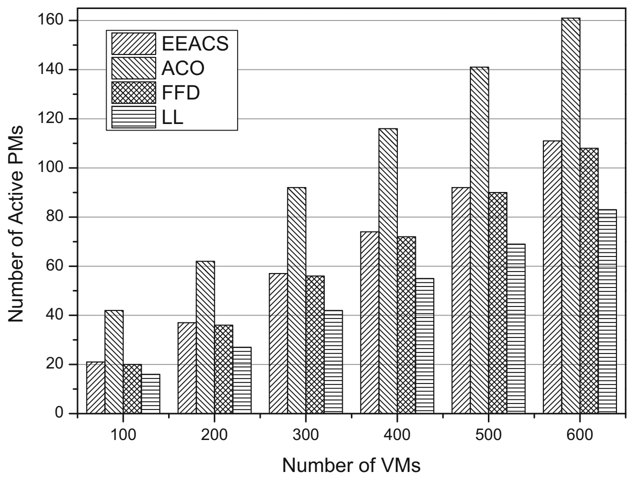

Fig. 7 Comparative results of number of active servers for four algorithms across six scenarios in a heterogeneous environment

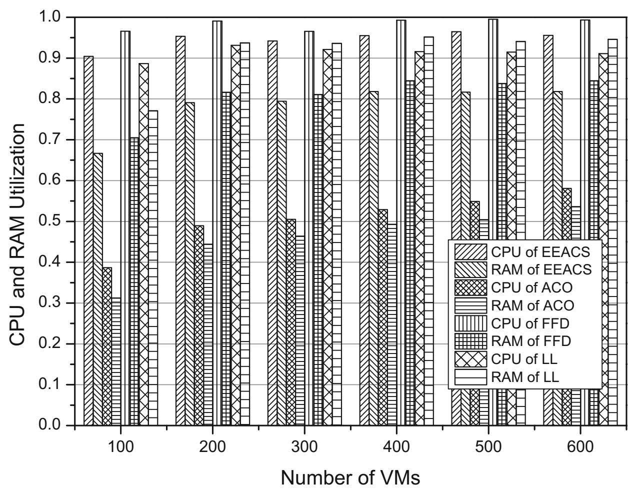

Fig. 8 Comparative results of CPU and RAM utilization among four algorithms across six scenarios in a heterogeneous environment

results. Specifically, the LL algorithm leads to the highest energy consumption for CDC to complete this batch of VM requests, with a 7.04% higher energy consumption than the FFD algorithm when the number of VM equals 600. The main reason for the higher energy consumption of the LL algorithm is that, although IBM servers offer higher energy efficiency, their resource capacity is indeed lower than DELL servers. Therefore, the LL algorithm tends to prioritize the larger-capacity but less energy-efficient DELL servers, whereas the FF algorithm adopts the opposite strategy. Among the four algorithms, EEACS achieves the best results in all six VM size scenarios. Specifically, when the number of VMs is 600, the EEACS algorithm reduces energy consumption by 10.49%, 54.38%, and 57.38% compared to the ACO, FFD, and LL algorithms, respectively.

Figure 7 shows the number of active servers for the four algorithms in six scenarios. The LL algorithm has the lowest number of active servers. When the number of VM is 600, the LL algorithm uses 25 fewer servers than the FFD algorithm. The FFD algorithm, which is a bin-packing approach aiming to minimize the number of PMs used. However, the FFD algorithm employs a higher number of PMs than the LL algorithm because it prioritizes IBM servers with higher energy efficiency but smaller capacity. Our EEACS algorithm's results are relatively close to the FFD algorithm. The ACO algorithm uses more active servers compared to the other three algorithms. When the number of VMs equals 600, the number of active servers used by the EEACS algorithm is 31.06% lower than that of the ACO algorithm.

Figure 8 compares the CPU and RAM utilization of the four algorithms. As illustrated in Fig. 8, the FFD algorithm achieves the highest CPU utilization, while the LL algorithm maintains a balanced utilization of both CPU and RAM. The ACO algorithm has the lowest CPU and RAM utilization. In terms of resource utilization, the EEACS algorithm is close to the FFD algorithm and significantly outperforms the ACO algorithm. In the six scenarios of the heterogeneous computing environment, the average CPU and RAM utilization of our EEACS algorithm are 86.69% and 70.88% higher than those of the ACO algorithm, respectively.

Table 3 Experimental result comparisons in both homogenous and heterogeneous environments

Environment	Metrics	Performance gain of EEACS (%)		
		FFD	LL	ACO
Homogenous	Energy	60.12	60.30	15.51
	Active PMs	− 4.59	− 4.23	43.51
	CPU utilization	− 4.89	− 3.88	80.39
	RAM utilization	− 4.84	− 3.97	79.84
Heterogeneous	Energy	53.72	57.03	14.19
	Active PMs	− 2.62	− 34.25	36.16
	CPU utilization	− 3.87	3.54	86.69
	RAM utilization	− 3.17	− 14.20	70.88

Table 3 presents the average performance improvement ratio of our algorithm's four evaluation metrics across six scenarios, compared to the other three algorithms. As shown in the table, the CPU and RAM utilization of EEACS is relatively worse than that of the FFD algorithm. However, the average decrease in utilization can be controlled within 5%, while the energy saving ratio reaches 60%. Compared with the data of the four indicators of the ACO algorithm, EEACS can save more than 10% of the electricity consumed, while simultaneously exhibiting a significant advantage over the ACO algorithm in terms of enhancing resource utilization. In summary, the experimental results demonstrate that our algorithm, in both homogeneous and heterogeneous cloud computing environments, can provide a reliable low-power consumption solution for large-scale VMP placement problems while maintaining resource utilization rates close to bin-packing algorithms.

5.3 Discussion on limitation

Although this study provides an effective solution to the large-scale VMP problem, our proposed algorithm still exhibits several limitations. Firstly, the resource requests of VMs should be expanded to encompass bandwidth and external storage resources in addition to CPU and RAM. With this expansion, the algorithm can incorporate mechanisms for balanced allocation across multiple dimensions of resources, aiming to achieve efficient and balanced resource utilization. Secondly, in this work, we employed a static VMP approach to allocate PMs for an existing set of VMs. We did not consider dynamic VMP and the resulting issues, such as VM migration and increased power consumption costs. However, to improve it, we can incorporate a cloud-edge-end computing architecture in our future work, investigating the time and energy costs associated with VM migrations, and focusing on minimizing the number of migrations and optimizing SLA violation. This approach can be compared with different dynamic VMP solutions, including those proposed in [4, 12, 13, 22]. Finally, we employed an ACS to minimize the number of active servers serving user demands while ensuring that these servers operate at higher efficiency load states. Although we evaluated our algorithm using four performance metrics, as it did not account for VM migrations during dynamic VM consolidation, future work will necessitate the incorporation of SLA indicators to compare and evaluate migration costs, similar to those presented in [4].

6 Conclusions

This paper proposes an energy-efficient VMP algorithm based on ACS, called EEACS. The algorithm utilizes an evolutionary approach to solve the NP-hard problem, aiming to optimize multiple objectives including energy consumption, resource utilization, and minimization of the number of active servers. We introduce the concept of an energy-consuming block, which represents the energy consumption incurred by placing a VM on a server and quantifies the impact of the VM on the energy consumption of the server. Using this concept, we define an energy consumption model for PM and specify the pheromone levels and heuristic information for ACS. When searching for a suitable server for a VM, we first identify all active servers that meet the resource requirements. If such servers exist, we select the most promising one based on pheromone and heuristic information. If no active servers satisfy the conditions, we choose the most energy-efficient inactive server, power it on, and assign it to the VM. This approach reduces the number of active servers, effectively narrowing the search scope and improving the speed of server selection. We conducted simulations of the VMP problem in both homogeneous and heterogeneous environments. The experimental results demonstrate that our algorithm significantly outperforms greedy algorithms (FFD and LL) and the evolutionary algorithm (ACO) in terms of energy savings. While there is no significant difference in the number of active servers and resource utilization compared to greedy algorithms, our approach is clearly superior to ACO. In conclusion, EEACS can significantly improve system energy efficiency while maintaining reasonable resource utilization. In the future, we plan to investigate the dynamic VMP multi-objective optimization problem within the context of a cloud-edge-end architecture.

Acknowledgements This research was funded by the Talent Cultivation and Teaching Reform Project of Chengdu University (Grant: cdjgb2022282), the Second Batch of Industry-University Cooperative Education Project, Ministry of Education, China (Grant: 202102123022), the Research Cultivation Project of School of Computer Science, Chengdu University (Grant: RCN_K016).

Author contributions Lin-Tao Duan proposed the improved ACS-based energy efficiency strategy (EEACS) for VMP problems and programmed the algorithm. Jin Wang analyzed and interpreted data. Hai-Ying Wang prepared the graphic materials. Lin-Tao Duan wrote the paper. All authors reviewed the paper.

Data availability No datasets were generated or analysed during the current study.

Declarations

Competing interests The authors declare no competing interests.

References

1. Buyya, R., Srirama, S.N., Casale, G., Calheiros, R., Simmhan, Y., Varghese, B., Gelenbe, E., Javadi, B., Vaquero, L.M., Netto, M.A.S., Toosi, A.N., Rodriguez, M.A., Llorente, I.M., Vimercati, S.D.C.D., Samarati, P., Milojicic, D., Varela, C., Bahsoon, R., Assuncao, M.D.D., Rana, O., Zhou, W., Jin, H., Gentzsch, W., Zomaya, A.Y., Shen, H.: A manifesto for future generation cloud computing: research directions for the next decade. ACM Comput. Surv. **51**, 5 (2018). https://doi.org/10.1145/3241737
2. Wan, B., Dang, J., Li, Z., Gong, H., Zhang, F., Oh, S.: Modeling analysis and cost-performance ratio optimization of virtual

machine scheduling in cloud computing. IEEE Trans. Parallel Distrib. Syst. **31**(7), 1518–1532 (2020). https://doi.org/10.1109/TPDS.2020.2968913
3. Ajiro, Y., Tanaka, A.: Improving packing algorithms for server consolidation. In: Proceedings International Conference Computer Measurement Group, pp. 399–406 (2007)
4. Ajmera, K., Tewari, T.K.: SR-PSO: server residual efficiency-aware particle swarm optimization for dynamic virtual machine scheduling. J. Supercomput. **79**(14), 15459–15495 (2023). https://doi.org/10.1007/s11227-023-05270-8
5. Shirvani, M.H., Seifhosseini, S.: Power management of cloud datacenter in infrastructure level via efficinet virtual machine placement by utilizing hybrid genetic algorithm. In: 2023 International Symposium on Signals, Circuits and Systems (ISSCS), pp. 1–4 (2023). https://doi.org/10.1109/ISSCS58449.2023.10190873
6. Feng, H., Deng, Y., Zhou, Y., Min, G.: Towards heat-recirculation-aware virtual machine placement in data centers. IEEE Trans. Netw. Serv. Manag. **19**(1), 256–270 (2022). https://doi.org/10.1109/TNSM.2021.3120295
7. Balaji, K., Sai Kiran, P., Sunil Kumar, M.: Power aware virtual machine placement in iaas cloud using discrete firefly algorithm. Appl. Nanosci. **13**(3), 2003–2011 (2023). https://doi.org/10.1007/s13204-021-02337-x
8. Feller, E., Rilling, L., Morin, C.: Energy-aware ant colony based workload placement in clouds. In: IEEE/ACM International Conference on Grid Computing (2011). https://doi.org/10.1109/Grid.2011.13
9. Xing, H., Zhu, J., Qu, R., Dai, P., Luo, S., Iqbal, M.A.: An aco for energy-efficient and traffic-aware virtual machine placement in cloud computing. Swarm Evol. Comput. **68**, 101012 (2022). https://doi.org/10.1016/j.swevo.2021.101012
10. Dorigo, M., Gambardella, L.: Ant colony system: a cooperative learning approach to the traveling salesman problem. IEEE Trans. Evol. Comput. **1**, 53–66 (1997). https://doi.org/10.1109/4235.585892
11. Liu, X.F., Zhan, Z.H., Deng, J., Li, Y., Gu, T., Zhang, J.: An energy efficient ant colony system for virtual machine placement in cloud computing. IEEE Trans. Evol. Comput. **22**, 113–128 (2016). https://doi.org/10.1109/TEVC.2016.2623803
12. Ajmera, K., Kumar Tewari, T.: Dynamic virtual machine scheduling using residual optimum power-efficiency in the cloud data center. Comput. J. **67**(3), 1099–1110 (2023). https://doi.org/10.1093/comjnl/bxad045
13. Ajmera, K., Tewari, T.K.: VMS-MCSA: virtual machine scheduling using modified clonal selection algorithm. Clust. Comput. **24**(3), 3531–3549 (2021). https://doi.org/10.1007/s10586-021-03320-5
14. Hsieh, S.Y., Liu, C.S., Buyya, R., Zomaya, A.Y.: Utilization-prediction-aware virtual machine consolidation approach for energy-efficient cloud data centers. J. Parallel Distrib. Comput. **139**, 99–109 (2020). https://doi.org/10.1016/j.jpdc.2019.12.014
15. Patel, N., Patel, H.: Energy efficient strategy for placement of virtual machines selected from underloaded servers in compute cloud. J. King Saud Univ. Comput. Inf. Sci. **32**(6), 700–708 (2020). https://doi.org/10.1016/j.jksuci.2017.11.003
16. Peake, J., Amos, M., Costen, N., Masala, G., Lloyd, H.: Paco-vmp: parallel ant colony optimization for virtual machine placement. Future Gener. Comput. Syst. **129**, 174–186 (2022). https://doi.org/10.1016/j.future.2021.11.019
17. Azizi, S., Shojafar, M., Abawajy, J., Buyya, R.: Grvmp: a greedy randomized algorithm for virtual machine placement in cloud data centers. IEEE Syst. J. **15**, 2571–2582 (2021). https://doi.org/10.1109/JSYST.2020.3002721
18. Zhao, C., Liu, J.: A virtual machine dynamic consolidation algorithm based dynamic complementation and ffd algorithm. In: 2015 Fifth International Conference on Communication Systems and Network Technologies, pp. 333–338 (2015). https://doi.org/10.1109/CSNT.2015.38
19. Feng, H., Deng, Y., Li, J.: A global-energy-aware virtual machine placement strategy for cloud data centers. J. Syst. Architect. **116**, 102048 (2021). https://doi.org/10.1016/j.sysarc.2021.102048
20. Zhao, H., Wang, J., Liu, F., Wang, Q., Zhang, W., Zheng, Q.: Power-aware and performance-guaranteed virtual machine placement in the cloud. IEEE Trans. Parallel Distrib. Syst. **29**(6), 1385–1400 (2018). https://doi.org/10.1109/TPDS.2018.2794369
21. Zhou, Z., Shojafar, M., Alazab, M., Abawajy, J., Li, F.: Afed-ef: an energy-efficient vm allocation algorithm for iot applications in a cloud data center. IEEE Trans. Green Commun. Netw. **5**(2), 658–669 (2021). https://doi.org/10.1109/TGCN.2021.3067309
22. Ma, X., Xu, H., Gao, H., Bian, M., Hussain, W.: Real-time virtual machine scheduling in industry IoT network: a reinforcement learning method. IEEE Trans. Ind. Inf. **19**(2), 2129–2139 (2023). https://doi.org/10.1109/TII.2022.3211622

Lin-Tao Duan received the BS degree in Information and Technology from the Central China Normal University, in 2001, and he received the MS degree in Computer Science from Chengdu University of Technology, in 2006. He got his Ph.D. in computer science from the School of Computer Science, Sichuan University in 2014. He is an associate professor of the School of Computer Science at the Chengdu University. His current research interests include embedded real-time system, cloud computing and green computing.

Jin Wang received his BS degree in Electronic Information System from Jilin University in 2003, and got the MS degree and Ph.D. degree from University of Electronic Science & Technology China in 2006 and 2013, respectively. He is an associate professor with Chengdu University. His current research mainly focuses in mobile edge computing and edge intelligence.

Hai-Ying Wang received the Ph.D. degree in computer application from the school of information science and technology, Southwest Jiaotong University. He is a lecturer in the School of Computer Science at Chengdu University. His research interests include network architecture, data mining and cloud computing.