A Virtual Machine Consolidation Algorithm Based on Dynamic Load Mean and Multi-Objective Optimization in Cloud Computing

Pingping Li * and Jiuxin Cao

Abstract: High energy consumption and low resource utilization have become increasingly prominent problems in cloud data centers. Virtual machine (VM) consolidation is the key technology to solve the problems. However, excessive VM consolidation may lead to service level agreement violations (SLAs). Most studies have focused on optimizing energy consumption and ignored other factors. An effective VM consolidation should comprehensively consider multiple factors, including the quality of service (QoS), energy consumption, resource utilization, migration overhead and network communication overhead, which is a multi-objective optimization problem. To solve the problems above, we propose a VM consolidation approach based on dynamic load mean and multi-objective optimization (DLMM-VMC), which aims to minimize power consumption, resources waste, migration overhead and network communication overhead while ensuring QoS. First, based on multi-dimensional resources consideration, the host load status is objectively evaluated by using the proposed host load detection algorithm based on the dynamic load mean to avoid an excessive VM consolidation. Then, the best solution is obtained based on the proposed multi-objective optimization model and optimized ant colony algorithm, so as to ensure the common interests of cloud service providers and users. Finally, the experimental results show that compared with the existing VM consolidation methods, our proposed algorithm has a significant improvement in the energy consumption, QoS, resources waste, SLAs, migration and network overhead.

Keywords: cloud computing; VM consolidation; energy saving; multiple objective optimization; load mean

1. Introduction

With the rapid development of cloud computing, the scale of data centers becomes larger and larger, and a large number of hosts around the world consume a huge amount of power every day, resulting in high CO₂ emissions [1]. Studies have shown that the average CPU utilization of physical hosts in cloud data centers is only 15–20%, and the energy consumed by an idle server is 70% of the peak energy [2,3], which not only wastes energy and resources but also greatly increases the operating costs of cloud service providers. In addition, VMs perform data intensive applications by accessing physical networks and communicating with interdependent VMs on different servers, which not only increases the network traffic but also reduces the overall performance of the data center’s network [4]. For users, this seriously hinders the customer’s experience, while for cloud service providers, a QoS may not be guaranteed, leading to SLAs. Therefore, considering the interests of cloud service providers and users, the above issues generate a complex and challenging multi-objective resource management problem in cloud computing.

The cloud computing resources are provided to users in the form of a VM. Therefore, the core problem of resource management in cloud computing is the VMs management, and VM consolidation technology is the main method used to solve this problem. The key to VM consolidation is to dynamically obtain the optimal mapping between VMs and hosts,
so as to minimize the energy consumption and resources waste to reduce the operating costs while ensuring a QoS for customers.

The main process of VM consolidation is to first determine the host load, then decide whether to migrate the VMs according to the host load status, and finally migrating the VMs to a new host. However, a dynamic cloud environment brings more challenges to an effective VM consolidation. First, in a dynamic cloud environment, especially in a cloud data center with thousands of hosts, detecting the host load state effectively and accurately has been an important problem to be solved. Some related studies [3,5,6] determine the host load status based on static thresholds, which lack the awareness of dynamic workloads in the data center. The current resources utilization cannot objectively and truly reflect the real load of hosts, which will lead to an excessive consolidation. The literature [7] proposes a dynamic threshold method, which considers the workload changes in the source host and target host after the VMs migration, but does not consider the resource load balance, which will lead to a large resources waste. In addition, some of the existing methods determine the host load state only based on the CPU resource [3–10], ignoring the impact of memory, network and other resources, which are also key factors affecting the QoS. Some studies [11–14] consider the impact of multi-dimensional resources, but mostly focus on the optimization of energy consumption and SLA, ignoring the VM migration overhead and the network communication overhead. For example, during the process of a VMs migration, the performance of the VM will be reduced, and placing two interdependent VMs at a long distance will cause a large communication delay [15].

VM consolidation is a well-known NP-hard problem [16]. Some studies have used heuristic algorithms to solve this question [3,17–19]. Heuristic algorithms are widely used in VM consolidation because of their simple implementation and low complexity. However, traditional heuristic algorithms are prone to fall into local optimal solutions. In the existing research, meta-heuristic algorithms are used to solve the VM placement problem [16,20–22], such as ant colony optimization (ACO), artificial bee colony, genetic algorithm, etc. These algorithms can effectively avoid local optimal solutions when solving large-scale model problems. The ant colony system (ACS) is a classical method of ACO. It can get a better solution for complex combinatorial problems in an appropriate time. Because of its excellent performance in solving NP-hard and combinatorial optimization problems [23,24], it has attracted more and more attention in solving VM consolidation problems. In this paper, the VM consolidation is considered to be a multi-objective combinatorial optimization problem. Therefore, the ACS is selected as the solution. However, with the increasing size of the data centers, some studies [25–27] using the ACS to solve the VM consolidation problem have resulted in too long an execution time due to the increasing search space of the ACS as well. Therefore, it is necessary to further optimize the search performance.

To solve the above problems, this paper proposes a VM consolidation algorithm based on the dynamic load mean and multi-objective optimization in cloud computing, in which the host load status is comprehensively measured based on multi-dimensional resources, and the dynamic characteristics of the system load are considered. Then, the optimized ant colony algorithm is used to obtain the optimal mapping between the VMs and the hosts so as to realize the multiple objectives optimization of the resource utilization, energy consumption, migration and communication overhead. Our main contributions are as follows:

1. A host load detection method based on the dynamic load mean is proposed which uses multi-dimensional resources to comprehensively measure the host load status and considers the impact of load fluctuations to avoid an excessive consolidation.
2. A network-aware model is proposed to optimize the network communication overhead of interdependent VMs and the overall network traffic of the data center.
3. An improved ant colony optimization algorithm is proposed to obtain a better solution and execution efficiency through the optimization of the heuristic factors and execution process.
The rest of the paper is organized as follows. Section 2 describes the related work. Section 3 introduces the relevant models. Section 4 introduces the host load detection method based on the dynamic load mean. Section 5 introduces the DLMM-VMC algorithm in detail. Section 6 gives the experimental results and performance evaluation. Section 7 concludes the paper and discusses the future work.

2. Related Work

A VM consolidation mainly solves three problems. That is, how to determine the host load status which mainly involves the host load detection methods, how to select the migration VMs which mainly involves the migration overhead calculation and how to select the placement hosts which mainly involves the best target host selection for the migration of the VMs.

For how to determine the host load status, some studies [3,5,6,28,29] used static thresholds of CPU utilization to determine whether a host was overloaded or underloaded, keeping the CPU utilization of the host between two fixed thresholds. However, in a cloud data center, resource utilization is constantly changing in multiple resource dimensions. In such as dynamic cloud environment, setting static thresholds or using the current utilization of a single resource is not an effective approach, which leads to an excessive consolidation. Therefore, some dynamic threshold algorithms were proposed. Beloglazov et al. [17] proposed an adaptive upper and lower thresholds method that sorted hosts based on the historical CPU data statistical analysis and CPU utilization prediction, which improves the ability to sense the dynamic changes in the host load. However, the load fluctuation was not considered. Chen et al. [30] proposed a host load detection method based on the time sliding window, which recorded the host CPU utilization in a certain time window through a regular sampling. When the host CPU utilization continuously exceeded the predefined threshold, the host was determined to be overloaded. Yadav et al. [31] proposed two adaptive methods based on robust regression to dynamically set the thresholds. Zhou et al. [32] proposed a dynamic adaptive three threshold host load detection method, using a K-Means clustering algorithm to divide the hosts into four types. However, these methods use CPU utilization as the main criterion to determine the host load. Therefore, it is not possible to accurately describe the load status of the hosts with multi-dimensional resource characteristics, which ultimately leads to an unnecessary migration and resources waste.

For how to select the migration of VMs, in [17,33], the authors proposed several migration VM selection strategies, which have been widely used. These strategies include the maximum utilization (MU) strategy, which selects a VM that has the highest CPU usage; the random selection (RS) strategy, which selects a VM randomly based on a uniformly distributed discrete random variable; the maximum correlation (MC) strategy, which selects a VM with the highest correlation with other VMs; and the minimum migration time (MMT) strategy, which selects a VM with the shortest migration time. Li et al. [34] used the similarity between memory contents of VMs to select a migration VM. This method aimed to select a VM with the highest similarity in memory contents from different hosts to reduce the migrated data and time. Masoumzadeh et al. [35] proposed a VM selection strategy based on fuzzy Q-learning, where multiple VM selection technologies were integrated, and the VM selection strategy was selected dynamically based on the fuzzy logic theory according to the current state of a host. Laili et al. [36] proposed a selection mechanism based on the iterative prediction algorithm, which used a reverse selection mechanism to select the most suitable VM from the candidate VM set for each randomly selected host. However, these studies only consider the migration overhead when selecting migration VMs, ignoring the impact on the host load. For overloaded hosts, the overload status should be quickly and accurately eliminated with the minimum migration overhead, while for low loaded hosts, the resources usage should be quickly reduced to shut down the host as soon as possible to reduce the energy consumption.

For how to select the placement hosts, the literatures [3,18] studied how to use the heuristic greedy algorithm to solve the VM placement problem. For instance, with the
first fit (FF), first fit decreasing (FFD), best fit (BF) and best fit decreasing (BFD) algorithms [30,31], which aim to reduce the number of running hosts and the VM migration number. However, classical heuristic algorithms are not convenient for VM consolidation, so many subsequent studies have improved these algorithms to make them applicable to VM consolidation. Beloglazov et al. [17] proposed a power-aware best-fit decreasing (PABFD) VM placement algorithm, which selects a host with the least energy consumption increase as a placement host. Li et al. [37] proposed virtual switch aware BFD and FFD algorithms, which comprehensively considered the traffic between VMs and the CPU overhead generated by virtual switches. Moges et al. [38] proposed a modified best fit decreasing (MBFD) algorithm to improve SLAv and active hosts. Zhang et al. [39] further optimized the MBFD algorithm by combining FF and MBFD to achieve a better energy efficiency.

The heuristic greedy algorithm has a low complexity, but it is not suitable for solving large-scale problems and cannot be well applied to large-scale data centers. The meta-heuristic algorithm has significant advantages in solving such problems. Li et al. [13] proposed a QoS-aware and multi-objective dynamic VM consolidation (QMOD) based on improved the genetic algorithm, which optimizes the three objectives of the load balancing, migration overhead and QoS. Li et al. [20] constructed the VM consolidation problem as a multi-objective optimization problem with multi resource constraints and solved the problem based on the artificial bee colony algorithm. They also proposed a VM consolidation method based on differential evolution (DE) [22]. However, in these methods, only the energy consumption and host overload risk are considered. Al-Moalmi et al. [40] proposed a VM placement method based on grey wolf optimization (GWO), which can use the CPU and RAM resources more effectively and reduce the number of active hosts, energy consumption and SLAv. Aryania et al. [26] proposed an energy-aware VM consolidation based on the ACS, which takes the energy consumption caused by VM migration as an important optimization goal. Farahnakian et al. [27] proposed an ACS-based VM consolidation approach that aims to maximize the number of dormant hosts and minimize the number of VM migrations. However, the above VM consolidation algorithms based on ant colony optimization take too long to execute for larger-scale data centers due to the large search space. Xiao et al. [25] proposed an improved ACS to solve the VM consolidation problem; they used the design heuristic factors to select both migration VMs and placement hosts and limited the search space of ants according to the type of host load state, thus reducing the blindness of the ant search and optimizing the execution efficiency. However, they ignore the optimization of selecting migration VMs, and the selection of migration VMs and placement hosts in VM consolidation jointly determines the search performance and execution efficiency of the ACS. Moreover, the above algorithms do not consider resources waste and network overhead. To some extent, the unbalanced resources utilization will greatly increase the resources waste, leading to an increase in the active hosts, thus increasing the energy consumption. In addition, the increase in the network overhead will greatly delay the response time of the application in the VM, which is easy to lead to a QoS degradation. Therefore, VM consolidation should comprehensively consider the interests of both the cloud service providers and the users to ensure that the cloud service providers minimize the operational costs while also ensuring a QoS for the users.

3. System Model
3.1. Data Center Resource Representation Model

Assume $H = \{h_1, h_2, \cdots, h_j, \cdots, h_n\}$ is the collection of a data center’s hosts, where $n$ is the number of hosts, each host has $D$-dimensional resources and $C_{h_j}^d, d \in \{1, 2, \cdots, D\}$ represents the capacity of resources $d$ of a host $h_j$. $V = \{v_1, v_2, \cdots, v_i, \cdots, v_m\}$ is the collection of the data center’s VMs, where $m$ is the number of VMs; similarly, each VM also has $D$-dimensional resources, and $C_{v_i}^d, d \in \{1, 2, \cdots, D\}$ represents the capacity of resources $d$ of a VM $v_i$. 
Further, assume $VM(h_j)$ indicates the VM collection of a host $h_j$, $U^d_t(v_i)$ represents the actual utilization of resource $d$ of a VM $v_i$ at time $t$, and $U^d_t(h_j)$ is the actual utilization of resource $d$ of a host $h_j$ at time $t$, which can be expressed as Equation (1).

\[
U^d_t(h_j) = \frac{1}{C^d_{h_j}} \sum_{v_i \in VM(h_j)} U^d_t(v_i) \times C^d_{v_i}
\]  \hspace{1cm} (1)

3.2. Energy Consumption Model

The energy consumption of a host in a data center is mainly composed of its related components CPU, memory, hard disk and network, but studies have shown that CPU is the main energy consuming device, and there is a linear relationship between CPU utilization and host energy consumption [3]. Therefore, we can establish the following energy consumption model, as shown below.

\[
P_j = k \times P_j^{max} + (1 - k) \times P_j^{max} \times U^d_t(h_j)
\]  \hspace{1cm} (2)

where $P_j^{max}$ is the energy consumption of a host $h_j$ when it is fully loaded, $U^d_t(h_j)$ is the cpu utilization of a host $h_j$ at time $t$, $k$ is the energy consumption factor and studies show that the energy consumption overhead when the hosts are idle is 70% of that when they are fully loaded, so $k$ is generally set to 0.7 [3]. In addition, through the above analysis, we can see that idle servers generate more energy consumption, and these hosts can be shut down in time to reduce the energy consumption.

Therefore, the total energy consumption in the data center is calculated as follows.

\[
P = \sum_{i=1}^{n} P_j
\]  \hspace{1cm} (3)

3.3. Resources Waste Model

Each host in the data center may run multiple VMs at the same time, and different VMs may run different applications, so the resources requirements in various dimensions are different. An unreasonable resources allocation will increase the resources waste. Therefore, it is important to ensure that the remaining resources of each dimension on the host are balanced in order to fully utilize the resources and prevent the resources waste. So, we propose a resources waste model, as shown below:

\[
W(h_j) = \sum_{d_x \in D, d_y \in D} \left| \left(1 - U^d_t(h_j)\right) - \left(1 - U^d_t(h_j)\right) \right|, d_x \neq d_y
\]  \hspace{1cm} (4)

\[
W = \sum_{j=1}^{n} W_j
\]  \hspace{1cm} (5)

where $U^d_t(h_j)$ and $U^d_t(h_j)$ represent the utilization of resources $d_x$ and $d_y$ on a host $h_j$, respectively. $W(h_j)$ and $W$ denote the resources waste of the host $h_j$ and the data center, respectively.

3.4. Communication Overhead Model

The more links between VMs which pass through, the greater the network latency, which is one of the most important factors affecting the QoS. Therefore, by optimizing the communication path between VMs, the network latency can be reduced. The total network communication overhead $N$ is calculated based on the network communication overhead between pairs of VMs and the distance to the physical hosts where they are located, as shown in the following equation.

\[
W = \sum_{j=1}^{n} W_j
\]  \hspace{1cm} (6)
where \( a(v_i, v_j) \) indicates the network communication overhead between \( v_i \) and \( v_j \), \( h(v_i) \) and \( h(v_j) \) denote the hosts where \( v_i \) and \( v_j \) are located, respectively, and \( b(h(v_i), h(v_j)) \) denotes the network communication distance between \( h(v_i) \) and \( h(v_j) \), which is measured using the number of switches and routes passed by both in the communication process; the larger the value, the greater the network communication overhead.

Placing pairs of VMs with a high network communication overhead on the same or closer hosts can reduce the network communication overhead and greatly reduce the communication latency.

3.5. Migration Overhead Model

A VM migration overhead is also a very important optimization objective, because VM migration consumes additional compute resources, and excessive VM migrations can also lead to large workloads and energy consumption. What’s worse, VM migration can degrade the QoS. Therefore, the number of VM migrations should be minimized during VM consolidation.

\[
M = \sum_{i=1}^{m} m(v_i), \forall m(v_i) = 1
\]

where \( m(v_i) = 1 \) indicates that VM \( v_i \) needs to be migrated and \( M \) indicates the total number of migration VMs.

3.6. Multi-Objective Optimization

Minimizing the energy consumption, network overhead, migration overhead and resources waste are the multiple optimization objectives to be obtained for VM consolidation in this paper. According to Equations (3) and (5)–(7), we get the following multi-objective optimization model with some constraints.

\[
\min F = \vartheta_1 P + \vartheta_2 W + \vartheta_3 N + \vartheta_4 M
\]

Constraints:

\[
x_{v_i h_j} = \begin{cases} 1, & \text{if } v_i \text{ is placed on host } h_j \\ 0, & \text{otherwise} \end{cases} \quad \forall v_i \in V \text{ and } \forall h_j \in H
\]

\[
\sum_{i=1}^{n} x_{v_i h_j} = 1, \forall h_j \in H
\]

\[
\sum_{i=1}^{n} C_{v_i}^{d} \times x_{v_i h_j} \leq C_{h_j}^{d}, \forall h_j \in H \text{ and } \forall d \in D
\]

\[
U_{d}^{d} (h_j) \leq Thr_{max}^{d} (h_j), \forall h_j \in H \text{ and } \forall d \in D
\]

where \( \vartheta_1 + \vartheta_2 + \vartheta_3 + \vartheta_4 = 1 \) and \( \vartheta_1, \vartheta_2, \vartheta_3, \vartheta_4 \) are the weight values. Constraint (10) ensures that the VM is allocated to only one host. Constraints (11) and (12) help ensure that each host meets the resource requirements of the VMs on it and does not exceed the maximum threshold \( Thr_{max}^{d} (h_j) \).

4. Host Load Detection Based on Dynamic Load Mean

4.1. Dynamic Load Mean

Host load detection is a key step in VM consolidation. Any kind of resource overload (e.g., CPU, memory, network and storage) will greatly degrade the service performance and lead to SLAs. Therefore, the host load should be detected based on multi-dimensional resources. When hosts are overloaded, VM consolidation is performed to limit the resources utilization within a certain range to avoid a performance degradation caused by a resource overload. In addition, the host load dynamically changes and fluctuates with its hosted VMs workload. A short-term fluctuation will not affect the performance of the system; if each fluctuation triggers the VMs migration, it will make the VMs consolidation too aggressive and bring a negative impact that not only poses the risk of a host overload, but also affects the performance of the applications on the VM.
In order to solve the above problems, we propose a host load detection method based on the dynamic load mean (DLM-HLD). On the one hand, the DLM-HLD uses multi-dimensional resources to calculate the comprehensive load of the hosts. On the other hand, it considers the impact brought by the system load’s fluctuation, and uses the dynamic load mean within the recent sliding time window when calculating the resource’s load in each dimension. The sliding time window size can be dynamically adjusted according to the load fluctuation’s size, thus dynamically adjusting the load mean. The host $h_j$ comprehensive load $L(h_j)$ is calculated as shown in Equation (13).

$$L(h_j) = \sum_{d \in D} \omega(d) \times U^d(h_j, T)$$  \hspace{1cm} (13)$$

where $\omega(d)$ is the weight coefficient of the resource $d$, and $U^d(h_j, T)$ represents the resource $d$ load mean in the sliding window $T$, which is calculated based on multiple discrete samples $U_r = \{U^d_{t_1}(h_j), U^d_{t_2}(h_j), \ldots, U^d_{t_i}(h_j), \ldots, U^d_{t_k}(h_j)\}$ in the sliding window time $T = \{t_1, t_2, \ldots, t_i, \ldots, t_k\}$, where $k = T/5$ takes a positive value and samples are collected every 5 s. The sliding time window size $T$ can be dynamically adjusted according to the absolute difference between the maximum value $\max(U_r)$ of the resource $d$ in time $T$ and the load mean $U^d(h_j, T - 1)$ in time $T - 1$, as shown in Equation (15). The larger the absolute difference, the larger the sliding time window size, and the larger the samples number, then the better ability to withstand load fluctuations during the VM consolidation. The parameter $s$ defines the sensitivity to changes in resource $d$. The smaller $s$ is, the more sensitive it is to perceive changes.

Different resources have a different utilization on a host. The higher the resource utilization, the greater the impact on the host overload and the higher the weight is. The information entropy algorithm determines the weight according to the variation degree of the metric. The greater the variation degree, the greater the impact, the smaller the information entropy and the smaller its weight. On the contrary, the smaller the variation degree, the smaller the impact, the larger the information entropy and the smaller its weight. In order to objectively perceive the comprehensive host load state, this paper uses information entropy to calculate the comprehensive host load $L(h_j)$, and the specific steps are as follows.

1) The decision matrix is calculated as the following equation, where each row of the matrix records the load mean of the resource $d$ corresponding to the host $h_j$, and each column of the matrix records the resource type.

$$U = \begin{bmatrix}
U^{d_1}(h_j, T) & \cdots & U^{d_d}(h_j, T) \\
\vdots & \ddots & \vdots \\
U^{d_1}(h_j, T) & \cdots & U^{d_d}(h_j, T)
\end{bmatrix}$$  \hspace{1cm} (16)$$

2) The matrix is normalized to obtain the matrix, as shown below.

$$R = \begin{bmatrix}
u_{11} & \cdots & u_{1d} \\
\vdots & \ddots & \vdots \\
u_{k1} & \cdots & u_{kd}
\end{bmatrix}$$  \hspace{1cm} (17)$$

where $u_{xy} = \frac{U^{d_{xy}}(h_j, T)}{\sum_{i=1}^k U^{d_{xy}}(h_j, T)}$.

3) Calculate the $y$-th term entropy $E_y = -\frac{1}{\ln k} \sum_{x=1}^k u_{xy} \ln u_{xy}$, $E_y \in [0, 1]$. 

(4) The computational contribution degree is \( d_y = 1 - E_y \), then the weight \( \omega_y \) is shown in the following equation.

\[
\omega_y = \frac{d_y}{\sum_{y=1}^{d} d_y}
\]

The \( \omega_y \) is the weight of the resource \( d_y \), and then the host \( h_j \) comprehensive load \( L(h_j) \) can be calculated by Equation (13).

4.2. Host Load Detection Base on Dynamic Load Mean

The pseudocode of the DLM-HLD is shown in Algorithm 1. First, we calculate the load mean \( U^d(h_j, T) \) of the host resource \( d \); then, the host comprehensive load \( L(h_j) \) is calculated based on \( U^d(h_j, T) \). Finally, based on the limit thresholds, all the hosts are divided into three categories: the overload, normal load and underload. Assume that the host comprehensive load upper and lower thresholds are \( Thr_{max} \) and \( Thr_{min} \), respectively, and the host resource \( d \) load upper and lower thresholds are \( Thr_{d, max} \) and \( Thr_{d, min} \), respectively. If \( L(h_j) > Thr_{max} \) or \( U^d(h_j, T) > Thr_{d, max} \), the host is classified as overload set \( H_o \); conversely, if \( L(h_j) \leq Thr_{min} \) or \( U^d(h_j, T) \leq Thr_{d, min} \), the host is classified as underload set \( H_u \). The remaining hosts are classified as the normal load set \( H_n \).

\[
\begin{align*}
\text{Algorithm 1 & Host load detection algorithm DLM-HLD} \\
\text{Input:} & \text{host list } H, \text{upper threshold } Thr_{max} \text{ and } Thr_{d, max}, \text{lower threshold } Thr_{min} \text{ and } Thr_{d, min} \\
\text{Output:} & \text{overloaded hosts } H_o, \text{normal hosts } H_n \text{ and underloaded hosts } H_u \\
1. & \text{Initialize: } H_o \leftarrow \emptyset, H_n \leftarrow \emptyset, H_u \leftarrow \emptyset \\
2. & \text{for } H_j \text{ in } H \text{ do} \\
3. & \quad \text{Base (15) compute } T \\
4. & \quad \text{Base (14) compute } U^d(h_j, T), t \in T \text{ and } U^d(h_j, t) \in Ur \\
5. & \quad \text{Base (16), (17), (18) compute } \omega(d) \\
6. & \quad \text{Base (13) compute } L(h_j) \\
7. & \quad \text{If } L(h_j) > Thr_{max} \text{ or } U^d(h_j, T) > Thr_{d, max} \text{ then} \\
8. & \quad \quad H_o \leftarrow H_j \\
9. & \quad \text{else if } L(h_j) \leq Thr_{min} \text{ or } U^d(h_j, T) \leq Thr_{d, min} \text{ then} \\
10. & \quad \quad H_u \leftarrow H_j \\
11. & \quad \text{else} \\
12. & \quad \quad H_n \leftarrow H_j \\
13. & \quad \text{End if} \\
14. & \text{End for} \\
15. & \text{Return } H_o, H_n, H_u 
\end{align*}
\]

5. The Proposed DLMM-VMC Algorithm

The following describes the main ideas of the DLMM-VMC. First, according to the DLM-HLD method, the hosts are divided into three categories: the overload, normal load and underload. Then, selecting the migration VMs and placement hosts based on the optimization ant colony algorithm. On the one hand, when selecting migration VMs, prioritize the VMs that make the greatest reduction in the overloaded resource utilization on the host, which effectively reduces the migration VMs number. In addition, to save energy, migrate as many VMs on underload hosts as possible to shut down more hosts. On the other hand, when selecting placement hosts, prioritize the hosts that make the best utilization of the resources on the host, which effectively reduces the resources waste. Finally, based on the multi-objective function proposed in Equation (8) and the optimized ant colony algorithm, the optimal solution is obtained. Define the mapping tuple set of migration VMs and placement hosts as \( TC = \{(v_m, h_p)\} \), where \( v_m \) is a VM to be migrated.
and $h_p$ is a placement host for migration VMs to be placed. The elements in the collection $TC$ are used as food for the ants. Ants search for solutions from $TC$ and use objective function (8) to evaluate the solutions, and finally get the optimal solution.

To reduce the time complexity of this method, we optimized the execution process of the ACS, as shown in Figure 1. This method restricts the solution search space of the ACS to a certain range of hosts instead of all the hosts based on the host load types output by the DLM-HLD algorithm. During the VM consolidation process, select the overloaded hosts and the underloaded hosts in turn, and when selecting the placement hosts for migration VMs, select the normal host, underloaded host and new host in turn. Therefore, compared with the original ant colony optimization algorithm, this method reduces the search range of the ants. In addition, to further optimize the execution efficiency for the VM consolidation problem, the heuristic factor takes into account both the migration VMs selection and placement hosts selection. On the one hand, the optimized heuristic factor selects a different migration VM selection heuristic factor based on the source host load type, which ensures minimizing the migration overhead when selecting the migration VMs. On the other hand, the optimized heuristic factor selects a different placement host selection heuristic factor based on the target host load type, which ensures a maximum resource utilization when selecting the placement hosts. The heuristic factor comprehensively considers the two key processes of the migration VMs selection and placement hosts selection, which helps to ensure the solution quality while reducing the blindness of the ant search to improve the execution efficiency of the ACS. In the following, detailed definitions of these factors are given.

![Figure 1. DLMM-VMC flow chart.](image-url)

**5.1. Pheromone Definition**

Pheromone $\tau_{ij}$ is the medium that ants communicate with each other. Ants find food sources by sensing other ants’ pheromones, and the higher the pheromone concentration, the greater the preference. Suppose that $\tau_{ij}$ denotes the pheromone on the combination $(v_i, h_j)$ of VM $v_i$ and host $h_j$, the pheromone $\tau_{ij}$ value changes due to the new pheromone accumulation and the old pheromone volatilization. The local pheromone update rule is as follows.

$$\tau_{ij} = (1 - \rho) \cdot \tau_{ij} + \rho \cdot \tau_0$$  \hspace{1cm} (19)

where $\rho \in [0, 1]$ is the pheromone volatility coefficient and $\tau_0$ is the initial pheromone that is a constant. Updating the local pheromone will reduce the pheromone concentration to avoid a premature convergence to suboptimal solutions.

After all the ants have constructed their solutions, the global optimal solution is obtained according to the objective function and the global pheromone is updated using
the global optimal solution to enhance the experience of the global optimal solution. The global pheromone update rule is as follows.

$$\tau_{ij} = (1 - \rho) \tau_{ij} + \rho \cdot F(X^+)$$  \hspace{1cm} (20)$$

where $X^+$ is the global optimal solution.

5.2. Definition of Heuristic Factor

In addition to the pheromone, the heuristic factor $\eta_{ij}$ is another very critical factor in an ant colony algorithm. The heuristic factor represents the expectation that the VM $v_i$ is assigned to host $h_i$. The larger the heuristic information is, the greater the corresponding behavior probability is. Therefore, a reasonable setting of the heuristic factor can reduce the search blindness and improve the search efficiency of the ant colony. This paper comprehensively considers the service performance and energy consumption, and sets the heuristic factor as shown in the following equation, which consists of two parts: the selection of migration VMs and placement hosts.

$$\eta_{ij} = \lambda \cdot \eta_v(h_i, v_i) + (1 - \lambda) \cdot \eta_h(h_j, +v_i), \lambda \in [0, 1]$$  \hspace{1cm} (21)$$

where $\eta_v(h_i, v_i)$ is the migration VMs selection heuristic factor, which indicates that the VM $v_i$ is migrated from the host $h_i$, and $\eta_h(h_j, +v_i)$ is the placement hosts selection heuristic factor, which indicates that the VM $v_i$ is migrated to the host $h_j$. $\lambda$ is the relative weight to measure the relative importance of the two. The settings of $\eta_v(h_i, v_i)$ and $\eta_h(h_j, +v_i)$ are described in detail below.

5.2.1. Migration VM Selection

For overloaded hosts, any kind of resource overload may affect the QoS and result in SLA. In addition, an improper policy for selecting migration VMs will cause too many VMs to be migrated, which increases the migration overhead. Therefore, for overloaded hosts, the main strategy for selecting a migration VM is to minimize the VM migration number and time under the premise of comprehensively considering multi-dimensional resources, and quickly restores the host from the overload state to the normal state. Therefore, we define the migration VM selection heuristic factor for the overloaded hosts as follows.

$$\eta_v(h_i, v_i) = \sum_{d \in D} \omega(d) \times \left( U_{\omega}^d(h_i) - U_{\omega}^d(h_i, v_i) \right) - T_{\text{mig}}(h_i, v_i)$$  \hspace{1cm} (22)$$

where $U_{\omega}^d(h_i)$ is the utilization of the resource $d$ of the overloaded host $h_i$, $U_{\omega}^d(h_i, v_i)$ is the utilization of the resource $d$ after the host $h_i$ migrating out of the VM $v_i$, $\sum_{d \in D} \omega(d) \times \left( U_{\omega}^d(h_i) - U_{\omega}^d(h_i, v_i) \right)$ is the load’s comprehensive descending gradient of the host $h_i$ after the VM $v_i$ is migrated out from the host $h_i$ and $\omega(d)$ is the weight value obtained based on Equation (18); the greater the descent gradient, the greater the probability of the VM $v_i$ being selected. Additionally, considering the migration time $T_{\text{mig}}(h_i, v_i)$ as a migration VM selection factor, this paper uses the migration time evaluation model proposed in the literature [41] to calculate $T_{\text{mig}}(h_i, v_i)$, which evaluates the VM migration time based on the current memory usage, dirty page and data transfer rate. $\eta_v(h_i, v_i)$ comprehensively considers the VM migration number and time. When selecting a migration VM, the faster the host overload state decreases and the shorter the migration time, the more likely the VM will be selected.

For underload hosts, in order to minimize the underload host number, preference is given to VMs that can significantly reduce the host’s resources utilization after migration to shut down the host. Therefore, we define the migration VM selection heuristic factor on the underload hosts as follows.

$$\eta_v(h_i, v_i) = \sum_{d \in D} \left( 1 - U_{\omega}^d(h_i, -v_i) \right)$$  \hspace{1cm} (23)$$
5.2.2. Placement Host Selection

When any resource usage of a host is overloaded, the host performance will drop rapidly. Therefore, when the normal load host is selected as the placement host, the one with more remaining resources is preferred. Additionally, consider resources waste and choose the one with the less resources waste. We comprehensively consider the QoS and resources waste and set the heuristic factor for normal load hosts in the following formula.

\[ \eta_h(h_j, +v_i) = |1 - L(h_j, +v_i) - W(h_j)| \]  

(24)

where \( 1 - L(h_j, +v_i) \) and \( W(h_j) \) are the remaining comprehensive load and resources waste of host \( h_j \) after deploying VM \( v_i \). The larger the \( \eta_h(h_j, +v_i) \), the greater the remaining comprehensive load, and the less resources waste there is.

For the underload hosts, their resources utilization are low and resources competition are weak, which can guarantee a QoS but cause a waste of resources and energy. Therefore, when selecting underload hosts as the placement hosts, the hosts with a higher resource utilization after deploying VMs are preferred to fully utilize the resources. The corresponding heuristic factor is defined as follows.

\[ \eta_h(h_j, +v_i) = |L(h_j, +v_i) - W(h_j)| \]  

(25)

5.3. Pseudo-Random Proportion Rule

According to the heuristic factor and pheromone information, the ants construct the solution according to the following pseudo-random proportion rule.

\[
(v_m, h_p) = \begin{cases} 
\arg \max \{ \frac{\tau^\alpha_{ij} \eta^\beta_{ij}}{\tau^\alpha_{ij} \eta^\beta_{ij}}, \text{if } q \leq q0 \} 
\end{cases}, \text{otherwise}
\]

(26)

where \( q \in [0, 1] \) is a uniformly distributed random number and \( q0 \in [0, 1] \) is a fixed parameter determining the relative importance of cumulative experience and random selection. The \( \alpha \) and \( \beta \) indicate the importance of the pheromone and heuristic factor. When \( q \leq q0 \) is called an exploitation, it is helping the ants to converge quickly to a high-quality solution, otherwise it is called an exploration, in which ants randomly select a tuple \((v_i, h_j)\) according to the probability distribution defined in the following equation, helping the ants to discover more new choices.

\[ p^k_{mp} = \begin{cases} 
\frac{\tau^\alpha_{ij} \eta^\beta_{ij}}{\sum_{(v, h) \in TC^k_{allow}} (\tau^\alpha_{ij} \eta^\beta_{ij})}, \text{if } (v_i, h_j) \in TC^k_{allow} 
0, \text{otherwise}
\end{cases} \]  

(27)

where \( TC^k_{allow} \) denotes the set of tuples that ant \( ant_k \) is allowed to traverse, and \( p^k_{mp} \) denotes the probability that ant \( ant_k \) selects the tuple \((v_m, h_p)\) next.

5.4. VM Consolidation Algorithm

The pseudo-code of the DLMM-VMC is shown in Algorithm 2. The input host sets are obtained by Algorithm 1. The algorithm initializes the iterations number \( nI \), ants number \( nA \) and \( \tau_0 \) (line 1). In each iteration, \( nA \) ants traverse the overloaded and underloaded hosts in turn, calculate heuristic factors based on the source and target host load types and reconstruct new mapping relationships between the VMs and hosts (lines 3–38). First, the VM consolidation is performed for the overloaded hosts (lines 4–20), and heuristic factors are calculated according to the overload host type (lines 7 and 11). Then, a VM consolidation is performed for the underload hosts (lines 21–36), and the heuristic factors are calculated according to the underload host type (lines 23 and 28). When selecting placement hosts, this algorithm selects the normal host and the underload host in turn (lines 5, 10, 22 and 27). After each ant constructs the solution, the local pheromone is updated (lines 19 and 35), and when all the ants have constructed the
while loop traverses the overloaded and underloaded hosts with the number of traversals of ants, \( m \) this algorithm is executed, \( \tau \) is updated using \( \tau \). When all the iterations are executed, \( X^+ \) is the final optimal solution.

**Algorithm 2 VM consolidation algorithm DLMM-VMC**

**Input:** overloaded host set \( H_o \), normal host set \( H_n \) and underloaded host set \( H_u \).

**Output:** \( X^+ \)

1. Initialize: \( nI, nA, \tau_0 \), \( X^+ \leftarrow \emptyset \), \( X^k \leftarrow \emptyset \) and \( M \leftarrow \emptyset \).
2. **for** \( i = 1 \) to \( nI \) **do**
3. **for** \( j = 1 \) to \( nA \) **do**
4. **while** \( H_o ! = \emptyset \) **do**
5. \( TC \leftarrow \{ (v_m, h_p) | v_m \in VM(h_j), h_j \in H_o \ and \ \forall h_p \in H_n \} \)
6. Based on Equations (21), (22) and (24), compute heuristic \( \eta_{mp} \) and \( \forall (v_m, h_p) \in TC \).
7. Based on Equation (27), compute \( p_{mp} \) and \( \forall (v_m, h_p) \in TC \).
8. Based on Equation (26), select \( (v_m, h_p) \in TC \).
9. **if** \( (v_m, h_p) \) is null, **then**
10. \( TC \leftarrow \{ (v_m, h_p) | v_m \in VM(h_j), h_j \in H_o \ and \ \forall h_p \in H_n \} \).
11. Based on Equations (21), (22) and (24), compute heuristic \( \eta_{mp} \), \( \forall (v_m, h_p) \in TC \).
12. Based on Equation (27), compute \( p_{mp} \) and \( \forall (v_m, h_p) \in TC \).
13. Based on Equation (26), select \( (v_m, h_p) \in TC \).
14. **if** \( (v_m, h_p) \) is null, **then**
15. Break
16. **end if**
17. **end if**
18. Update mapping relation matrix \( X \).
19. Update local pheromone using (19).
20. **end while**
21. **while** \( H_u ! = \emptyset \) **do**
22. \( TC \leftarrow \{ (v_m, h_p) | v_m \in VM(h_j), h_j \in H_u \ and \ \forall h_p \in H_n \} \)
23. Based on Equations (21), (23) and (24), compute heuristic \( \eta_{mp} \) and \( \forall (v_m, h_p) \in TC \).
24. Based on Equation (27), compute \( p_{mp} \) and \( \forall (v_m, h_p) \in TC \).
25. Based on Equation (26), select \( (v_m, h_p) \in TC \).
26. **if** \( (v_m, h_p) \) is null, **then**
27. \( TC \leftarrow \{ (v_m, h_p) | v_m \in VM(h_j), h_j \in H_u \ and \ \forall h_p \in H_n \}, j \neq d \)
28. Based on Equations (21), (23) and (25), compute heuristic \( \eta_{mp} \) and \( \forall (v_m, h_p) \in TC \).
29. Based on Equation (26), select \( (v_m, h_p) \in TC \).
30. **if** \( (v_m, h_p) \) is null, **then**
31. Break
32. **end if**
33. **end if**
34. Update mapping relation matrix \( X \).
35. Update local pheromone using (19).
36. **end while**
37. \( M \leftarrow M \cup \{ X^k \} \)
38. **end for**
39. \( X^+ \leftarrow \arg \max_{X^k \in M} \{ F(X^k) \} \).
40. Based on Equation (20), update global pheromone using \( X^+ \).
41. **end for**

As shown in Algorithm 2, we can conclude that the maximum time complexity of this algorithm is \( O(nI \cdot nA \cdot m \cdot n) \). Where \( nI \) is the number of iterations, \( nA \) is the number of ants, \( m \) is the number of VMs and \( n \) is the number of hosts. In line 4 and line 21, the while loop traverses the overloaded and underloaded hosts with the number of traversals
less than \( n \), and then in line 5, line 10, line 22 and line 27, the VMs on the overloaded and underloaded hosts are traversed in turn to construct the solution space with the number of traversals less than \( m \cdot n \). Because we handle overloaded hosts and underloaded hosts separately, select normal load hosts and underloaded hosts sequentially when selecting the placement hosts. Therefore, the number of VMs and hosts traversed each time is less than \( m \) and \( n \), respectively, and the final complexity of the DLMM-VMC is less than or equal to \( O(nI \cdot nA \cdot m \cdot n) \).

6. Performance Analysis and Discussion

6.1. Experimental Setup

The proposed algorithm was evaluated using the simulator CloudSim [30], which is a cloud computing environment simulation framework that can simulate most of the resources and behaviors of the cloud systems.

This experiment simulated a cloud data center with two types of hosts. The host types are HP ProLiant G4 and ProLiant G5, and the details of their configurations are shown in Table 1 and the energy consumption characteristics are shown in Table 2. The hosts were connected through a gigabit network. Four types of Amazon EC2 VMs [17] were used, and their configuration information is shown in Table 3. After the VM instances were created, they were initially deployed based on the resource requirements of the VM type.

Table 1. Host Configuration Information.

| Types   | CPU Type        | Frequency (GHz) | Core | RAM(GB) |
|---------|-----------------|-----------------|------|---------|
| HP ProLiant G4 | Inter Xeon 3040 | 1.86            | 2    | 4       |
| HP ProLiant G5 | Inter Xeon 3075 | 2.66            | 2    | 4       |

Table 2. Energy consumption of experimental host (watt).

| Types | 0%  | 10% | 20% | 30% | 40% | 50% | 60% | 70% | 80% | 90% | 100% |
|-------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|------|
| HP G4 | 86  | 89.4| 92.6| 96  | 99.5| 102 | 106 | 108 | 112 | 114 | 117  |
| HP G5 | 93.7| 97  | 101 | 105 | 110 | 116 | 121 | 125 | 129 | 133 | 135  |

Table 3. VM configuration information.

| Types                  | CPU Frequency (MIPS) | RAM(GB) |
|------------------------|----------------------|---------|
| High-CPU medium instance | 2500                | 0.85    |
| Extra large instance   | 2000                | 3.75    |
| Small instance         | 1000                | 1.7     |
| Micro instance         | 500                 | 0.613   |

To verify the validity of the proposed algorithm, the real-world workload dataset of the Google cluster data (GCD) [42] was used in the experiment. The GCD provided real tracking data for approximately a month in May 2011, which was tracked every five minutes and tracked multiple resources utilization, such as the CPU and memory. The data of different days were randomly selected from the processed data. The statistical characteristics of the 1600 VMs critical resources are shown in Table 4. The algorithm related parameter settings are shown in Table 5. Here, we set \( \theta_i = 0.25 \) and \( d = 2 \), which indicates that the multiple objectives have the same weight and consider using the CPU and memory resources.
The service level agreement (SLA) refers to an agreement reached between the cloud service providers and users on services, priorities and responsibilities. If the SLA is violated, the users’ interests cannot be guaranteed, and the cloud service providers may pay expensive fines to users as compensation. Therefore, the SLA is an important metric to measure a data center’s QoS. SLAv \[ 17\] are an independent metric to measure SLA violations, which is measured from two aspects: the SLA violation time caused by the host overload (SLAHv) and the performance degradation caused by the VM migration (SLAMv). These two aspects are independent and have the same impact on SLAv. Therefore, the total SLAv are calculated as follows:

$$\text{SLAv} = \text{SLAHv} \times \text{SLAMv}$$  \hspace{1cm} (28)

The SLAHv indicates the percentage of time when the CPU or memory usage of a host reached 100%; meanwhile, the SLAMv indicate the overall performance degradation caused by the VM migration. The SLAHv and SLAMv values are calculated as follows:

$$\text{SLAHv} = \frac{1}{n} \sum_{j=1}^{n} \frac{T_{h_j}}{T_{a_j}}$$  \hspace{1cm} (29)

$$\text{SLAMv} = \frac{1}{m} \sum_{i=1}^{m} \frac{C_{v_i}}{C_{a_i}}$$  \hspace{1cm} (30)

where \(n\) and \(m\) indicate the numbers of hosts and VMs in a data center, respectively; \(T_{h_j}\) and \(T_{a_j}\) represent the number of time when the host utilization reached 100% and the total running time, respectively; \(C_{v_i}\) represents the capacity of the unfulfilled resource requests caused by the VM migration, which is the estimation of the performance degradation caused by the VM migration; and \(C_{a_i}\) is the total CPU requirement for a VM \(v_i\) during its lifetime. Studies have shown that \[17\] SLAMv can be set to 10% of the CPU utilization during the VM migration.

The energy consumption in the VM consolidation is an important evaluation metric. However, when optimizing the energy consumption, SLAv need to be balanced. The comprehensive evaluation metric PSV is calculated from the combination of the total energy consumption and SLAv, and is defined as follows.

$$\text{PSV} = P \times \text{SLAv}$$  \hspace{1cm} (31)

where \(P\) is the total energy consumption according to Equation (3); when the PSV value is low, it indicates that the data center has a good performance in terms of the energy consumption and QoS.

In addition, the network communication overhead and migration overhead are also the optimization objectives in this paper, which are evaluated based on Equations (6) and (7), respectively. The network communication overhead \(a(v_i, v_j)\) between VMs \(v_i\) and \(v_j\) is calculated by referring to the literature \[43\].
6.3. Performance of DLM-HLD

In this experiment, data centers of different sizes were used to evaluate the performance of the DLM-HLD algorithm. The number of hosts in a data center varied from 100 to 1500, and each host was initially deployed with two VMs on average. The experiment focuses on the impact of the dynamic load mean on the overloaded, underloaded, active hosts number and the migration number in different sizes of data centers. The DLM-HLD scheme was compared with the static and dynamic threshold detection methods proposed in [17]. The static threshold detection method (THR-HLD) set the maximum utilization of the CPU and memory to 80%. The dynamic threshold detection method (LR-HLD) estimated the threshold using the local regression (LR) method and detected overloaded hosts according to the estimated CPU and memory utilization values. The VM consolidation test was performed every 5 min, and the test results were recorded over 24 h. Figure 2 shows the test results.

![Figure 2.](image)

According to Figure 2a,b, compared with the THR-HLD and LR-HLD algorithms, the DLM-HLD algorithm proposed in this paper detected the least number of overloaded or underloaded hosts in the VM consolidation. For a 1500-node data center, the DLM-HLD detected 56.8% and 40.6% fewer overloaded hosts and 58.9% and 59.5% fewer underloaded hosts compared to the THR-HLD and LR-HLD, respectively. The THR-HLD algorithm is based on the current resources utilization as the criterion for detecting a host overload or underload, without considering the dynamic changes in the resources load. As long as the current resource utilization exceeds the set threshold, the host is judged as an overload or underload, and even occasional load fluctuations will detect the hosts overload or underload, which leads to the misjudgment of the hosts overload or underload in the VM consolidation, thus increasing the number of overloaded or underload hosts. Although the LR-HLD can predict a future resources utilization, it cannot predict occasional load fluctuations. The DLM-HLD algorithm considers the dynamic load mean of resources over a period, which not only accurately judges the trend of resources usage but also filters out occasional load fluctuations, thus effectively reducing the misjudgment in the host load detection.
Next, the impact of the dynamic load mean (DLM) on the number of active hosts in different sizes of data centers was analyzed. As shown in Figure 2c, the number of hosts that need to be activated using the DLM-HLD method was the smallest. From the above analysis, we know that the THR-HLD and LR-HLD algorithms detected more overloaded and underloaded hosts. However, each VM consolidation required a VMs migration on the overloaded and underloaded hosts, which led to the migration of more VMs, of which Figure 2d shows the result, and finally more hosts were activated when more migration VMs were placed.

6.4. Multi-Objective Optimization Performance

This section evaluated the multi-objective optimization performance of the DLMM-VMC. Four heuristic algorithms and two meta-heuristic algorithms were used as the comparison benchmarks. For heuristic algorithms, host load detection used two algorithms: the static threshold THR and dynamic threshold LR [17]. The migration VMs selection used the minimum migration time algorithm MMT [17], and the placement host selection used both the FF [3] and PABFD [17] algorithms. The two meta-heuristic algorithms were ACS-VMC [40] and QMOD [13], respectively. The data center was sized to deploy 400, 800 and 1200 VMs based on 400 physical servers for testing. The VM consolidation experiments were executed every 5 min, and the test results were recorded over 24 h, as shown in Figure 3.

![Figure 3. The comparison of (a) energy consumption and (b) resources waste.](image)

Figure 3a shows the energy consumption comparison. For the data center with 1200 VMs, the DLMM-VMC reduced the energy consumption by 30.8%, 27.7%, 30.3%, 19.1%, 23.5%, 19.3% and 9.8% compared with the THR-MMT-FF, THR-MMT-PABFD, LR-MMT-PABFD, ACS-VMC and QMOD, respectively. On the one hand, energy consumption was a major optimization objective in the DLMM-VMC, and the DLM-DLH method effectively reduced the number of active hosts. In addition, the resources waste was also our optimization objective. Figure 3b shows that the DLMM-VMC method has the least resources waste, which proves that the DLMM-VMC makes full use of resources so that it can minimize the active hosts’ number when deploying the same number of VMs. On the other hand, the DLMM-VMC optimized the heuristic factor of the ant colony algorithm. When selecting the placement hosts, the optimization heuristic factor fully considered the comprehensive resources utilization and resources waste of the host and selected the host with less resources waste under constraints. Therefore, compared with other algorithms, the DLMM-VMC effectively reduced the energy consumption.

Figure 4a illustrates the SLAv comparison. The results show that the DLMM-VMC has the best performance in SLAv, followed by the QMOD, and the THR-MMT-PABFD has the worst performance. For the data center with 1200 VMs, the DLMM-VMC SLAv were 73.6% of the QMOD, but was only 24.1% of the LR-MMT-PABFD, which proved that the DLMM-VMC algorithm effectively guaranteed the QoS. The SLAv were composed of SLAHv and SLAMv. In order to analyze the SLAv in more detail, we further analyzed the SLAHv and SLAMv.
overloaded hosts were quickly and accurately restored to a normal load level with a minimal migration overhead based on the optimized heuristic factors when selecting migration VMs. Therefore, compared with other comparison algorithms, the DLMM-VMC had obvious advantages in SLAv.

Figure 5a shows the network overhead comparison. Based on the tree network topology, the results show that the DLMM-VMC has the minimum network overhead, which proves that the network overhead model proposed in this paper effectively reduces the network communication cost. The DLMM-VMC placed the interdependent VMs close to each other so as to reduce the number of network elements that pass through during the network’s communication. It is well known that transmission information is processed and forwarded as it passes through the network’s elements, which increases the corresponding transmission delay. If VMs that communicate with each other are maximally deployed on the same server, the communication traffic handled by the network elements in the data center is greatly reduced, which not only reduces the overhead of the network resources but also improves the overall communication performance of the data center.

Figure 5b shows the comprehensive performance comparison. The results show that the DLMM-VMC has the lowest value, indicating that its comprehensive performance is the best. The PSV was composed of the total energy consumption $P$ and SLAv. The above analysis shows that $P$ and SLAv achieve the optimal results compared to other algorithms. Therefore, the PSV is also optimal.
overloaded hosts were quickly and accurately restored to a normal...the three meta-heuristic algorithms. However, the DLMM-VMC was better than the other two meta-heuristics and was close to the three heuristics. The DLMM-VMC algorithm limited the solution search space of the ant colony based on the host load types, which effectively improved the execution efficiency.

**Figure 5.** The comparison of (a) network overhead and (b) PSV.

### 6.5. Execution Efficiency Analysis

In order to deeply analyze the efficiency of the DLMM-VMC, the execution time was analyzed, as shown in Figure 6a. Due to the low time complexity, the four heuristic algorithms were shorter than the three meta-heuristic algorithms. However, the DLMM-VMC was better than the other two meta-heuristics and was close to the three heuristics. The DLMM-VMC algorithm limited the solution search space of the ant colony based on the host load types, which effectively improved the execution efficiency.

**Figure 6.** The comparison of (a) execution time and (b) convergence performance.

In addition, we compared the DLMM-VMC with the ACS-VMC in terms of the convergence. We calculated the objective function value according to Equation (8) and run the two algorithms 10 times separately. The number of VMs was set to 400. As seen in Figure 6b, both the DLMM-VMC and ACS-VMC converged in 500 iterations, and the DLMM-VMC solution was smaller. The DLMM-VMC algorithm converges significantly faster than the ACS-VMC algorithm, which starts to converge after 150 iterations, and the ACS-VMC algorithm has a convergence trend after 260 iterations. It can be seen that the DLMM-VMC has been improved in terms of the algorithm convergence performance.

### 7. Conclusions

This paper focuses on how to optimize the energy consumption, resource utilization, QoS, migration overhead and network communication overhead in cloud data centers, and thus proposes a DLMM-VMC algorithm to do so. The DLMM-VMC constructs the VM consolidation problem as a multiple-objective optimization problem. First, a host load detection method based on the dynamic load mean is proposed to objectively and accurately evaluate the real load state of the hosts, which avoids the deficiency of only considering single-dimensional resources in VM consolidation and also optimizes the problem of unnecessary VM migrations caused by system load fluctuations. Then, the optimized ant colony algorithm is proposed to obtain the optimal mapping scheme between the hosts and the VMs. In this process, the heuristic factor and the execution process of the ACS are optimized to achieve the improvement in the multiple objective optimization and execution efficiency. Finally, the experimental results show that the DLMM-VMC is effective in reducing the energy consumption, optimizing resources utilization, guaranteeing a QoS...
and reducing a migration overhead and network communication overhead compared with other algorithms.

This paper ignores the energy consumption generated by other devices in the data center and the impact on the system’s performance, such as the network elements and refrigeration equipment. In the future, we will comprehensively consider various factors to conduct VM consolidation research to further optimize the energy consumption.

Author Contributions: Conceptualization, P.L. and J.C.; methodology, P.L.; software, P.L.; validation, P.L. and J.C.; formal analysis, P.L.; investigation, P.L.; resources, P.L.; data curation, P.L.; writing—original draft preparation, P.L.; writing—review and editing, P.L. and J.C.; visualization, P.L.; supervision, J.C.; project administration, J.C.; funding acquisition, J.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China, grant number 62172089.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data will be available upon request through correspondence email.

Acknowledgments: The authors would like to thank the anonymous referees for their valuable comments and the support of the Southeast University.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Biswas, N.K.; Banerjee, S.; Biswas, U.; Ghosh, U. An approach towards development of new linear regression prediction model for reduced energy consumption and SLA violation in the domain of green cloud computing. *Sustain. Energy Technol. Assess.* 2021, 45, 101087. [CrossRef]

2. Birke, R.; Chen, L.Y.; Smirni, E. Data centers in the cloud: A large scale performance study. In Proceedings of the 5th IEEE International Conference on Cloud Computing, Honolulu, HI, USA, 24–29 June 2012; IEEE: Piscataway, NJ, USA, 2012; pp. 336–343.

3. Beloglazov, A.; Abawajy, J.; Buyya, R. Energy-aware resource allocation heuristics for efficient management of data centers for Cloud computing. *Future Gener. Comput. Syst.* 2012, 28, 755–768. [CrossRef]

4. Ardagna, D.; Casale, G.; Ciavotta, M.; Pérez, J.F.; Wang, W. Quality-of-service in cloud computing: Modeling techniques and their applications. *J. Internet Serv. Appl.* 2014, 5, 11. [CrossRef]

5. Haghshenas, K.; Pahlevan, A.; Zapater, M.; Mohammadi, S.; Atienza, D. MAGNETIC: Multi-Agent Machine Learning-Based Approach for Energy Efficient Dynamic Consolidation in Data Centers. *IEEE Trans. Serv. Comput.* 2022, 15, 30–44. [CrossRef]

6. Saadi, Y.; El Kafhali, S. Energy-efficient strategy for virtual machine consolidation in cloud environment. *Soft Comput.* 2020, 24, 14845–14859. [CrossRef]

7. Verma, A.; Dasgupta, G.; Nayak, T.K.; De, P.; Kothari, R. Server Workload Analysis for Power Minimization Using Consolidation. In Proceedings of the 2009 Conference on USENIX Annual Technical Conference, Berkeley, CA, USA, 14–19 June 2009; p. 28.

8. Wang, B.; Liu, F.; Lin, W. Energy-efficient VM scheduling based on deep reinforcement learning. *Future Gener. Comput. Syst.* 2021, 125, 616–628. [CrossRef]

9. Guazzzone, M.; Anglano, C.; Canonico, M. Exploiting VM Migration for the Automated Power and Performance Management of Green Cloud Computing Systems. In *International Workshop on Energy Efficient Data Centers*; Springer: Berlin/Heidelberg, Berlin, 2012; pp. 81–92. [CrossRef]

10. Feller, E.; Rilling, L.; Morin, C. Energy-aware ant colony based workload placement in clouds. In Proceedings of the IEEE/ACM International Conference on Grid Computing, Lyon, France; IEEE: Piscataway, NJ, USA, 2011; pp. 26–33.

11. Arshad, U.; Aleem, M.; Srivastava, G.; Lin, J.C.-W. Utilizing power consumption and SLA violations using dynamic VM consolidation in cloud data centers. *Renew. Sustain. Energy Rev.* 2022, 167, 112782. [CrossRef]

12. Kanagasubaraja, S.; Hema, M.; Valarmathi, K.; Kumar, N.; Kumar, B.P.M.; Balaji, N. Energy Optimization Algorithm to Reduce Power Consumption in Cloud Data Center. In Proceedings of the 2022 International Conference on Advances in Computing, Communication and Applied Informatics (ACCAI), Chennai, India, 28–29 January 2022; pp. 1–8. [CrossRef]

13. Li, J.; Zhang, R.; Zheng, Y. QoS-aware and multi-objective virtual machine dynamic scheduling for big data centers in clouds. *Soft Comput.* 2022, 26, 10239–10252. [CrossRef]

14. Sayadnavard, M.H.; Haghighat, A.T.; Rahmani, A.M. A multi-objective approach for energy-efficient and reliable dynamic VM consolidation in cloud data centers. *Eng. Sci. Technol. Int. J.* 2021, 26, 100995. [CrossRef]
15. Li, Z.; Yan, C.; Yu, X.; Yu, N. Bayesian network-based Virtual Machines consolidation method. *Future Gener. Comput. Syst.* 2017, 69, 75–87. [CrossRef]

16. Karmakar, K.; Das, R.K.; Khatua, S. An ACO-based multi-objective optimization for cooperating VM placement in cloud data center. *J. Supercomput.* 2021, 78, 3093–3121. [CrossRef]

17. Beloglazov, A.; Buyya, R. Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers. *Concurr. Comput. Pract. Exp.* 2011, 24, 1397–1420. [CrossRef]

18. Speitkamp, B.; Bichler, M. A Mathematical Programming Approach for Server Consolidation Problems in Virtualized Data Centers. *IEEE Trans. Serv. Comput.* 2010, 3, 266–278. [CrossRef]

19. Murtazaev, A.; Oh, S. Sercon: Server Consolidation Algorithm using Live Migration of Virtual Machines for Green Computing. *ITE Technol. Rev.* 2011, 28, 212–231. [CrossRef]

20. Li, Z.; Yu, X.; Yu, L.; Yu, X. Energy-aware and multi-resource overload probability constraint-based virtual machine dynamic consolidation method. *Future Gener. Comput. Syst.* 2018, 80, 139–156. [CrossRef]

21. Peake, J.; Amos, M.; Costen, N.; Masala, G.; Lloyd, H. PACO-VMP: Parallel Ant Colony Optimization for Virtual Machine Placement. *Future Gener. Comput. Syst.* 2021, 129, 174–186. [CrossRef]

22. Masoodzadeh, S.S.; Hlavacs, H. Integrating VM selection criteria in distributed dynamic VM consolidation using Fuzzy Q-

23. Li, Z.; Yan, C.; Yu, L.; Yu, X. Energy-efficient and quality-aware VM consolidation method. *Future Gener. Comput. Syst.* 2018, 80, 139–156. [CrossRef]

24. Abbass, H.; Greenwood, G.; Petraki, E. The SNs -Player Trust Game and its Replicator Dynamics. *IEEE Trans. Evol. Comput.* 2015, 20, 165–179. [CrossRef]

25. Abbas, H.; Greenwood, G.; Petkari, E. The SNs -Player Trust Game and its Replicator Dynamics. *IEEE Trans. Evol. Comput.* 2015, 20, 470–474. [CrossRef]

26. Farahnakian, F.; Ashraf, A.; Pahikkala, T.; Liljeberg, P.; Plosila, J.; Porres, I.; Tenhunen, H. Using Ant Colony System to Consolidate VMs for Green Cloud Computing. *IEEE Trans. Serv. Comput.* 2015, 8, 187–198. [CrossRef]

27. Mastroianni, C.; Meo, M.; Papuzzo, G. Probabilistic Consolidation of Virtual Machines in Self-Organizing Cloud Data Centers. *IEEE Trans. Cloud Comput.* 2013, 1, 215–228. [CrossRef]

28. Xiao, H.; Hu, Z.; Li, K. Multi-Objective VM Consolidation Based on Thresholds and Ant Colony System in Cloud Computing. *IEEE Access* 2019, 7, 53441–53453. [CrossRef]

29. Aryania, A.; Aghdasi, H.S.; Khani, L.M. Energy-Aware Virtual Machine Consolidation Algorithm Based on Ant Colony System. *J. Grid Comput.* 2018, 16, 477–491. [CrossRef]

30. Yadav, R.; Zhang, W.; Kaviwartya, O.; Singh, P.R.; Elgendy, I.A.; Tian, Y.-C. Adaptive Energy-Aware Algorithms for Minimizing Energy Consumption and SLA Violation in Cloud Computing. *IEEE Access* 2018, 6, 59923–59936. [CrossRef]

31. Zhou, Z.; Abawajy, J.; Chowdhury, M.; Hu, Z.; Li, K.; Cheng, H.; Alelaiwi, A.A.; Li, F. Minimizing SLA violation and power consumption in Cloud data centers using adaptive energy-aware algorithms. *Future Gener. Comput. Syst.* 2018, 86, 836–850. [CrossRef]

32. Calheiros, R.; Ranjan, R.; Beloglazov, A.; De Rose, C.A.F.; Buyya, R. CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Softw. Pract. Exp.* 2011, 41, 23–50. [CrossRef]

33. Li, H.; Li, W.; Wang, H.; Wang, J. An optimization of virtual machine selection and placement by using memory content similarity for server consolidation in cloud. *Future Gener. Comput. Syst.* 2018, 84, 98–107. [CrossRef]

34. Masoumzadeh, S.S.; Hlavacs, H. Integrating VM selection criteria in distributed dynamic VM consolidation using Fuzzy Q-Learning. In Proceedings of the 9th International Conference on Network and Service Management (CNSM 2013), Zurich, Switzerland, 14–18 October 2013; pp. 332–338. [CrossRef]

35. Laili, Y.; Tao, F.; Wang, F.; Zhang, L.; Lin, T. An Iterative Genetic Algorithm for Dynamic Virtual Machine Consolidation under Cloud Computing Environment (revised December 2017). *IEEE Trans. Serv. Comput.* 2021, 14, 30–43. [CrossRef]

36. Moges, F.F.; Abebe, S.L. Energy-aware VM placement algorithms for the OpenStack Neat consolidation framework. *J. Cloud Comput.* 2019, 8, 2. [CrossRef]

37. Zhang, X.; Wu, T.; Chen, M.; Wei, T.; Zhou, J.; Hu, S.; Buyya, R. Energy-aware virtual machine allocation for cloud with resource reservation. *J. Syst. Softw.* 2018, 147, 147–161. [CrossRef]

38. Al-Moalmi, A.; Luo, J.; Salah, A.; Li, K. Optimal Virtual Machine Placement Based on Grey Wolf Optimization. *Electronics* 2019, 8, 283. [CrossRef]

39. Liu, H.; Jin, H.; Xu, C.-Z.; Liao, X. Performance and energy modeling for live migration of virtual machines. *Clust. Comput.* 2013, 16, 249–264. [CrossRef]

40. Google Cluster Data. 2011. Available online: https://code.google.com/p/googleclusterdata/ (accessed on 17 July 2022).

41. Meng, X.; Pappas, V.; Zhang, L. Improving the Scalability of Data Center Networks with Traffic-aware Virtual Machine Placement. In Proceedings of the 2010 Proceedings IEEE INFOCOM, San Diego, CA, USA, 14–19 March 2010; pp. 1–9. [CrossRef]