A Dynamic Virtual Machine Resource Consolidation Strategy Based on a Gray Model and Improved Discrete Particle Swarm Optimization

YIFAN SHAO,1, QIANGQIANG YANG1,2, YAJUAN GU3, YONG PAN3, YI ZHOU4, AND ZIAO ZHOU5

1School of Electronics and Information Engineering, Shanghai University of Electric Power, Shanghai 201306, China
2School of Communication and Information Engineering, Shanghai University, Shanghai 200444, China
3Shanghai Rules and Research Institute, China Classification Society, Shanghai 200135, China
4MXSUN, Guangzhou 510285, China
5School of Electronic Information Science and Technology, Kunming University of Science and Technology, Kunming 650000, China

Corresponding author: Qiangqiang Yang (yangqiangqiang@shiep.edu.cn)

ABSTRACT
With the continuous expansion of the scale of cloud datacenters, high energy consumption has become a problem. Virtual machine (VM) consolidation is an effective method of energy management, but overly aggressive consolidation has caused issues such as high service level agreement violation (SLA) rates and excessive migrations. To reduce energy consumption while ensuring the quality of service of a datacenter, a VM dynamic consolidation strategy based on a gray model and an improved discrete particle swarm algorithm (GM-DPSO) is proposed. This strategy uses a gray prediction algorithm for load detection and adjusts the size of the threshold according to the load condition. Then, we select VMs for overloaded and underloaded hosts and complete the placement of VMs through a VM placement strategy based on an improved discrete particle swarm algorithm. This method aims to improve the load balance, establish a mapping between VMs and hosts, use particles to search for the best VM placement in the global scope, and reduce the probability of SLA. Our algorithm considers the impact of VM migration on system energy consumption and service quality and takes sufficient measures to reduce the number of migrations. To verify the effectiveness and practicality of this strategy, experiments were conducted on actual workloads, and the results were compared with those of other strategies. The experimental results show that the GM-DPSO greatly improves service quality while reducing energy consumption and SLA by 34.53% and 97.53%, respectively. The methods and theories in this paper provide strong theoretical and practical engineering guidance for large-scale cloud deployment.

INDEX TERMS
Dynamic VM consolidation, energy awareness, service quality, gray model, discrete particle swarm optimization.

I. INTRODUCTION
With the continuous development of cloud computing technology, the scale of cloud datacenters is expanding. Around the globe, 80% of the companies that deploy OpenStack cloud platforms have 100 or more cloud computing servers. The construction of cloud datacenters has increased energy consumption, and the insufficient utilization of cloud servers has further increased the operating costs. Therefore, the efficient use of datacenters has become an inevitable issue in the era of green and intelligent power consumption. According to statistics, datacenters that do not adopt effective energy-saving measures in China cover an area of 1,000 square meters, and they consume more than 500 kWh of electricity each year. Today, the CPU utilization of most resource center servers is less than 30%, and this usage increases energy consumption and resource waste to some degree [1]. In today’s datacenters, cloud providers need to reduce energy consumption to decrease operating costs, and one of the most effective ways is to formulate a virtual machine (VM) consolidation...
strategy to improve the utilization of dynamic resources [2]–[6]. However, the consolidation of VMs will inevitably lead to a decline in service quality, resulting in Service Level Agreement (SLA) violations. Therefore, when formulating a consolidation strategy, it is necessary to ensure excellent user service quality [7]–[10] while minimizing energy consumption.

VM consolidation is a dynamic method that decreases the number of hosts operating in a datacenter through VM migrations to gain the full benefit of computing resources and reduce energy consumption [11], [12]. Traditional VM consolidation methods tend to focus on energy consumption [13], [14] and use aggressive consolidation methods to shut down hosts, but this approach can seriously affect the operating performance of the system. With the continuous development of cloud computing technology in everyday applications, users are becoming more concerned with the stability of the cloud platform service quality. Generally, declines in service quality are mainly due to host overload and excessive VM migrations [10]. Therefore, we consider the service quality of a datacenter from the following perspectives: 1. Reasonable consolidation timing. The consolidation timing needs to be adapted to the utilization ability of the host, and a reasonable consolidation timing can improve the overall performance of the system under different load conditions [8]; 2. Decrease the number of VM migrations [16], [17]. A decrease in the number of VM migrations can significantly mitigate service quality degradation caused by migration; 3. Consider the load balance when placing VMs. Load balancing can balance the remaining resource space of each node, improve the scalability of the system, and reduce the probability of host overload.

The timing of the consolidation of VMs is determined by the load of the host. The load of the host often changes dynamically; if load changes were not considered, invalid migration could occur. In this paper, a gray prediction algorithm based on residual correction is used for load prediction. This algorithm can establish a nonlinear prediction model based on historical load data [19], and the prediction accuracy is higher than that of linear models [8], [20]. Traditional consolidation algorithms [6], [15] are often based on greedy models, and the algorithm complexity is low; however, it is difficult to obtain the global optimal solution with these models. The particle swarm optimization algorithm is a metaheuristic algorithm that can search for a solution that meets the requirements in the global scope and optimizes some aspects of the placement strategy approach [21]. The particle swarm optimization algorithm is usually used to solve continuous problems, so it needs to be coded to solve discrete problems. In this article, we will improve the particle swarm algorithm to adapt to the VM consolidation problem.

According to the above analysis, this paper proposes a VM dynamic consolidation strategy called GM-DPSO based on a gray model and improved discrete particle swarm optimization. We predict the future CPU utilization of the host by using the gray model to adaptively determine the consolidation time. During the process of VM consolidation, we select destination hosts through improved discrete particle swarm optimization to improve load balancing among hosts. The algorithm searches for the optimal placement strategy based on the changes in particle speed and position to avoid a result that becomes trapped in a local optimal solution. The work performed in this article is as follows:

1. We use dual thresholds for load detection and introduce gray prediction and dynamic threshold adjustment to optimize host overload detection and reduce the migration of VMs.
2. A VM selection strategy based on the migration time and CPU utilization is proposed to decrease the migration time and frequency.
3. A VM placement strategy based on an improved discrete particle swarm optimization is proposed to balance the placement of migrated VMs and achieve overall service optimization.
4. We evaluate the proposed consolidation strategy with real workloads from PlanetLab and compare it with other strategies based on different metrics.

The remaining sections of this article are as follows. Section 2 summarizes the current related work. Section 3 presents the system energy consumption model and the definition of each parameter. In Section 4, we introduce the proposed VM dynamic consolidation strategy. The experimental setup and the analysis of the experimental results are shown in Section 5. In Section 6, we summarize the article.

II. RELATED WORK
The dynamic consolidation of VMs can generally be divided into three parts: host load detection, the selection of VMs to be migrated, and the placement of VMs.

In the host load detection stage, the consolidation time needs to be determined based on the current load conditions. Beloglazov and Buyya [15] proposed five types of host overload detection: static threshold (THR), interquartile range (IQR), median absolute deviation (MAD), local regression (LR) and local robust regression (LRR) detection. The static threshold method uses a unified standard to handle overloaded hosts and cannot adapt to the current load changes. The other four algorithms can adaptively adjust the threshold according to historical load data to determine the migration timing for different load conditions. Fard et al. [22] used a temperature threshold to determine whether the host was overloaded. This algorithm monitors the temperature of all hosts and migrates VMs from overheated hosts to reduce the temperature. This algorithm can effectively control the performance and temperature of the host, but its disadvantage is the same as that of the traditional static threshold method; that is, it cannot effectively deal with load fluctuations. Yadav et al. [23] proposed an adaptive CPU utilization overload detection method based on Pearson’s correlation coefficient. In the algorithm, the higher the level of resource utilization by VMs is, the higher the probability of host
overloading. Prediction models play an important role in load detection. A predictive model can predict near-instantaneous changes in the load. By analyzing the current and future loads, we can determine the most reasonable migration time and reduce the number of invalid migrations. Li et al. [8] proposed a host overload detection method based on a robust linear regression prediction model, which enabled the host to complete the migration of VMs in advance before overloading. Compared with traditional linear regression models, several error correction methods are proposed for the model in this paper to improve the accuracy of the prediction results, and a safety factor is used to further reduce the SLA V rate. Hieu et al. [24] used linear functions for multiresource prediction at various times to estimate long-term utilization. Estimates of the long-term estimated resource utilization rate can be used to effectively perform overload and underload detection so that decision making can optimize system performance in a long period of time. Liu et al. [16] applied the autoregressive integrated moving average (ARIMA) model, which transforms an original time series into a stationary time series to predict the future host CPU utilization. Liu et al. [17] proposed an extreme learning machine for load prediction to establish a nonlinear model and predict changes in datacenter resources. Compared with linear programming prediction, this method can more accurately reflect variational trends.

In terms of VM selection, Beloglazov et al. [15] proposed four VM selection algorithms: minimum migration time (MMT), maximum correlation (MC), minimum utilization (MU), and random selection (RS) algorithms. During the process of VM migration, the service quality will be negatively affected. The MMT algorithm can reduce the time of VM migration and limit the performance degradation caused by migration. The MC algorithm considers the correlations among resources used by host applications. The higher the correlation is, the higher the probability of host overload will be. Therefore, the VM with the highest correlation with other VMs is selected for migration. Ammar et al. [7] wanted to maintain a balance among various host resources after VM selection, and they found that load-balanced hosts can accommodate more VMs than imbalanced hosts to prevent resource waste. Liu et al. [16] suggested that an overloaded host will cause SLAVs regardless of whether VM migration is performed. Therefore, a VM with a SLAV rate caused by migration that is less than the SLAV rate caused by overload should be selected for migration. This algorithm can find a balance between the two violations and reduce the overall violation level. Li et al. [25] considered the impact of VMs on host resources when selecting VMs. They selected the VM that contributed the most to the host load for migration, and the memory of the VM to be migrated was minimized.

The placement of VMs is similar to the multidimensional bin-packing problem. In [4], [8], [15], [24], [26]–[28], the authors proposed VM placement strategies based on a common bin-packing algorithm called BFD (best-fit decreasing algorithm). The BFD algorithm allocates VMs to the most suitable hosts in a specific order to minimize the consolidation time and complexity. In [4] and [15], the authors modified the BFD algorithm and proposed a power-aware BFD (PABFD) algorithm. PABFD selects the host with the smallest increase in energy consumption after allocation. Li et al. [17] proposed a VM placement strategy based on the resource capacity and energy consumption; specifically, the host with the largest resource capacity and lowest energy consumption is selected in each step as the target host. Hieu et al. [24] improved the PABFD algorithm proposed in [14] by adding a multitarget prediction module to reduce the possibility of host overload after VM migration. Li et al. [27] considered the impact of VM consolidation on VM switch traffic. The consolidation of VMs with excessively high sharing of virtual resources will limit the resources allocated by the VMs and reduce the performance of the VMs. Therefore, virtual switching overhead can be considered to effectively optimize the quality of service of VMs. Mastroianni et al. [30] established allocation functions for the CPU and memory resources of hosts. When a host has considerable remaining resources, they are most likely to be selected as the target host for VMs. Xiao et al. [31] merged VMs based on alliance games. The algorithm divides hosts into ultrahigh-load hosts, high-load hosts and low-load hosts according to the utilization rate and redistributes them by splitting the VMs for ultrahigh-load hosts, combining VMs for low-load hosts, and combining VMs for low-load hosts and high-load hosts. This method can ensure that the utilization of each host is within a reasonable range during allocation, avoid the excessive utilization of a single host, and effectively limit SLAVs caused by the excessively aggressive traditional consolidation algorithm. Li et al. [25] used a constructed Bayesian network to predict the probability of a VM being migrated when the VM was deployed on a physical host with different load patterns and further estimated the total number of potential VM migrations on the physical host. When consolidating VMs, VMs were migrated to physical hosts with few potential migrations and sufficient available resources. This method considers the possible occurrence of SLAVs at future time points during consolidation, and it can maintain the stability of the system over a long period of time. Liu et al. [17], Feller et al. [32], and Xiao et al. [33] all adopted metaheuristic algorithms based on ant colonies for VM consolidation. A metaheuristic algorithm can effectively avoid the local optimal solution and optimize the overall result. Compared with the greedy algorithm, the performance of the metaheuristic algorithm is superior. Ferdaus et al. [13] used the ant colony algorithm to place VMs on the least-utilized host and improved utilization by balancing resources among hosts. This method can effectively reduce
the energy consumption of datacenters but does not consider the quality of service, leading to SLAVs. Farahnakian et al. [14] proposed a dynamic consolidation algorithm for VMs based on ACS. The algorithm uses energy consumption, migration quantity and SLAVs metrics for multiobjective optimization. However, in the actual consolidation process, resulting in ineffective reduction of the number of migrations and an excessively long consolidation time for large datacenters. Aryania et al. [34] made improvements to the approach of Farahnakian et al. [14]. They considered the number of sleeping hosts and the energy consumption of VMs during the migration process when performing VM consolidation. Additionally, the impact of memory on energy consumption during migration was accounted for. According to the experimental results, the algorithm in [34] performed better than the algorithm in [14] based on energy consumption, the SLAV and other metrics. Wu et al. [35] combined a greedy heuristic algorithm and a genetic algorithm and proposed an improved-grouping genetic algorithm consolidation strategy to reduce energy consumption and migration costs. Gamal et al. [18] used a hybrid artificial bee colony and ant colony algorithm to optimize the dynamic environment of cloud computing. The experimental results verified that this strategy effectively balanced the load of each server, reduced energy consumption and the number of VM migrations, and improved service quality. Sharma and Reddy [36] used a hybrid genetic and particle swarm optimization algorithm to place VMs, and it reduced energy consumption and resource waste. The advantage of this method is that when the solution space is large and the adaptability level is low, the convergence speed of the genetic algorithm is slow. Notably, using the particle swarm algorithm to improve the adaptability of chromosomes can effectively reduce the time of VM consolidation and improve the consolidation quality. Liu et al. [37] considered the coordination problem for multiple VMs in a parallel application and proposed an initial virtual cluster allocation algorithm and a PM selection strategy based on PSO to reduce the overall transmission overhead, overall network resource consumption and total execution time for parallel applications. Wang et al. [38] imitated symbiotic survival and established a VM consolidation model. Biological competition at a same host will affect the host, and parasites will select a host with sufficient resources. This model can be well adapted to the consolidation of VMs. VMs at the same host are highly correlated and easily overloaded, and the performance of VMs can be ensured at hosts with sufficient resources. This article establishes a fitness function based on host utilization and the correlation of VMs and uses a biological heuristic algorithm to help VMs migrate to a suitable target host. Experiments have verified that this model can adapt to different datacenter loads and significantly reduce datacenter energy consumption and SLA violation rates. Ibrahim et al. [21] used the particle swarm algorithm to solve the problem of VM consolidation. The particle swarm algorithm can overcome continuity problems, so discretization is needed when using the algorithm to solve the box-packing problem. Discretization can lead to an insufficient number of particles in the population, and the algorithm may fall into a local optimal solution. Compared with [21], this paper adds a mutation subalgorithm to the algorithm, which mitigates the particle abundance issue. In addition, we introduce a local optimization algorithm to improve the particle movement direction.

III. ENERGY CONSUMPTION MODEL AND RELATED PARAMETERS

A. ENERGY CONSUMPTION MODEL

The energy consumption model of the host is mainly determined by the energy consumption of the CPU, memory [10], disk and VM migrations [39], [40]. The energy consumption model is as follows:

\[ E = E_{CPU} + E_{mem} + E_{disk} + E_{migration} \] (1)

where \( E_{CPU} \), \( E_{mem} \), \( E_{disk} \) and \( E_{migration} \) are the energy consumption levels of the CPU, memory, disk and VM migrations, respectively. The energy consumption of a single host is mainly determined by the CPU, and the large-scale migration of VMs in a datacenter requires additional migration energy. Therefore, the energy consumption model in this paper only considers \( E_{CPU} \) and \( E_{migration} \).

The CPU energy consumption model can be represented as follows:

\[ E_{CPU} = \int_{t}^{t+\Delta t} (P_{CPUi} + (P_{CPUm} - P_{CPUi}) \times U_{CPU}(t)) \, dt \] (2)

where \( P_{CPUi} \) and \( P_{CPUm} \) are the power consumption levels of the CPU for idle and full loads, respectively, and \( U_{CPU}(t) \) is CPU utilization.

The energy consumption associated with VM migration mainly includes the energy consumption of the host and network bandwidth during migration [41], and the model can be expressed as:

\[ E_{mig} = \sum_{i=1}^{N} (P_{host} + P_{bw}) \times T_{migi} \] (3)

where \( P_{host} \) is the maximum energy consumption of the source host and the target host during the VM migration process, \( P_{bw} \) is the network bandwidth energy consumption, \( T_{migi} \) is the time required to migrate VM i, and \( N \) is the total number of VMs to migrate. Migration energy consumption is mainly influenced by the increase in host energy consumption during the migration process; therefore, in this article, we only consider \( P_{host} \). In addition, according to the model, it can be concluded that the migration time and the number of migrations have a considerable impact on the total energy consumption of migration and need to be considered during consolidation.

B. PARAMETERS

For a cloud datacenter, \( HL = \{h_1, h_2, \ldots, h_n\} \) represents a list of hosts at the datacenter, and
VL_M_i = \{v_m_1, v_m_2, \ldots, v_m_N\} represents a list of VMs among hosts. Host resources (HR) and VM resources (VMR) can be defined considering the CPU, memory, network bandwidth and storage characteristics, where \( HR = \{CPU, RAM, BW, STORAGE\} \) and \( VMR = \{CPU_v, RAM_v, BW, STORAGE_v\} \).

The CPU utilization of the host in this paper is defined as follows:

\[
U_{CPU} = \sum_{i=1}^{n} U_{iCPU} \times \frac{CPU_{vi}}{CPU_h} \tag{4}
\]

where \( U_{iCPU} \) is the CPU utilization of VM_i at the host, \( CPU_{vi} \) is the MIPS allocated to VM_i, and \( CPU_h \) is the MIPS of the host.

The memory utilization of the host is defined as follows:

\[
U_{RAM} = \sum_{i=1}^{n} U_{iRAM} \times \frac{RAM_{vi}}{RAM_h} \tag{5}
\]

where \( U_{iRAM} \) is the memory utilization of the VM_i, \( RAM_h \) is the allocated memory capacity of the VM_i, and \( RAM_h \) is the memory capacity of the host.

The network bandwidth utilization of the host is defined as follows:

\[
U_{BW} = \frac{BW_{used}}{BW_h} \tag{6}
\]

where \( BW_{used} \) is the network bandwidth used by the host and \( BW_h \) is the total network bandwidth of the host.

The storage utilization of the host is defined as follows:

\[
U_{STO} = \frac{STORAGE_{used}}{STORAGE_h} \tag{7}
\]

where \( STORAGE_{used} \) is the used storage capacity of the host and \( STORAGE_h \) is the total storage capacity of the host.

C. MIGRATION COST

During the process of the live migration of VMs, although the VMs are not suspended, live migration has a negative impact on performance. According to the research in [10], the average performance degradation associated with VMs during the migration process is approximately 10% of the CPU utilization rate. Each migration may cause SLAVs and affect the operating performance of the system. Therefore, it is important to reduce the migration time and minimize the number of VM migrations. The VM migration cost model is as follows:

\[
T_{mig_i} = \frac{RAM_{vi}}{BW_h} \tag{8}
\]

\[
U_{deg_i} = 0.1 \cdot \int_{t_0}^{t_{0}+T_{mig_i}} u_i(t)dt \tag{9}
\]

where \( T_{mig_i} \) is the time required for the migration of VM_i, \( RAM_{vi} \) is the memory used by VM_i, \( BW_h \) is the available network bandwidth of the host, \( U_{deg_i} \) is the overall performance degradation associated with VM_i, and \( u_i(t) \) is the CPU utilization of VM_i.

IV. DYNAMIC MIGRATION STRATEGY OF VMs BASED ON A GRAY MODEL AND IMPROVED DISCRETE PARTICLE SWARM OPTIMIZATION

The proposed strategy can be divided into three parts: host load detection, the selection of VMs to be migrated, and VM placement. During host load detection, we set high and low thresholds with a gray prediction algorithm and dynamic threshold adjustment strategy to identify overloaded and underloaded hosts. In the process of VM selection, VMs are selected according to the proposed VM selection algorithm based on the MMT and CPU utilization. In the process of VM placement, considering load balancing among datacenter components, we adopt a VM placement strategy based on improved discrete particle swarm optimization to balance each host and improve system performance.

A. HOST LOAD DETECTION

The energy consumption and performance of a cloud datacenter are closely related to the load conditions. If the load is too high, insufficient resource space will be available on the host, which may cause SLAVs and affect the performance of VMs. If the load is too low, the resources on the host may not be effectively used, resulting in excess energy consumption. In this paper, we set the high threshold \( (T_{high}) \) and low threshold \( (T_{low}) \) of CPU utilization as the migration thresholds. If the CPU utilization level is higher than the high threshold, the host is overloaded, and some VMs must be migrated; if the CPU utilization level is lower than the low threshold, the host is underloaded, and all VMs need to be migrated. After migration, a sleep state can be used to reduce the number of hosts running at the datacenter.

During the host load detection process, due to the dynamic fluctuations in the load, a host may be overloaded, but the load will return to normal relatively quickly. This issue may cause excess migrations or invalid migrations, resulting in increases in energy consumption and migration costs. To solve this problem, we propose a gray prediction algorithm with residual correction to predict the loads of hosts in the future [19], [42]. The gray prediction algorithm is an exponential prediction model that uses a small amount of incomplete information to establish a mathematical model and make predictions. The algorithm predicts future CPU utilization based on historical load data. The process of the gray prediction algorithm is as follows:

First, we define the sequence:

\[
x^{(0)} = \{x^{(0)}(1), x^{(0)}(2), \ldots, x^{(0)}(N)\} \tag{10}
\]

\[
x^{(1)} = \left\{ \sum_{j=1}^{i} x^{(0)}(j) | i = 1, \ldots, N \right\} \tag{11}
\]

where \( x^{(0)} \) represents the historical CPU utilization data for the host.

GM (1,1) model:

The gray system is a differential equation established for discrete sequences, and GM (1,1) is a first-order differential
equation model. The form of GM (1,1) is as follows:

$$\frac{dx^{(1)}}{dt} + ax^{(1)} = b$$  \hspace{1cm} (12)

Discrete forms of the differential equations are established:

$$Z^{(1)}(k) = \mu \times x^{(1)}(k) + (1 - \mu) \times x^{(1)}(k - 1), \quad k = 2, 3, \ldots, n$$  \hspace{1cm} (13)

$$x^{(1)}(k) - x^{(1)}(k - 1) + aZ^{(1)}(k) = b$$  \hspace{1cm} (14)

In general, the value of \( \mu \) is 0.5:

$$Z^{(1)}(k) = 0.5x^{(1)}(k) + 0.5x^{(1)}(k - 1)$$  \hspace{1cm} (15)

Next, (13) is transformed into a matrix expression:

$$\begin{bmatrix} x^{(0)}(2) \\ x^{(0)}(3) \\ \vdots \\ x^{(0)}(N) \end{bmatrix} = \begin{bmatrix} -Z^{(1)}(2) & 1 \\ -Z^{(1)}(3) & 1 \\ \vdots & \vdots \\ -Z^{(1)}(n) & 1 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix}$$  \hspace{1cm} (16)

This equation can be rewritten as

$$y = (x^{(0)}(2), x^{(0)}(3), \ldots, x^{(0)}(N))^T$$  \hspace{1cm} (17)

$$B = \begin{bmatrix} -Z^{(1)}(2) & 1 \\ -Z^{(1)}(3) & 1 \\ \vdots & \vdots \\ -Z^{(1)}(n) & 1 \end{bmatrix}, \quad U = \begin{bmatrix} a \\ b \end{bmatrix}$$  \hspace{1cm} (18)

$$y = BU$$  \hspace{1cm} (19)

$$U = \begin{bmatrix} a \\ b \end{bmatrix} = (B^T B)^{-1}B^T y$$  \hspace{1cm} (20)

The prediction formula can then be obtained based on the above formula:

$$\hat{x}^{(1)}(k + 1) = [x^{(0)}(1) - \frac{b}{a}e^{-ak} + \frac{b}{a}, \quad k = 1, 2, \ldots, N$$  \hspace{1cm} (21)

$$\hat{x}^{(0)}(k) = \hat{x}^{(1)}(k) - \hat{x}^{(1)}(k - 1), \quad k = 2, 3, \ldots, N + 1$$  \hspace{1cm} (22)

To improve the accuracy of prediction, we use residuals to establish the GM (1,1) model and modify the prediction results. A residual is defined as follows:

$$E^{(0)}(k) = x^{(0)}(k) - \hat{x}^{(0)}(k)$$  \hspace{1cm} (23)

The GM (1,1) model for residuals is

$$\dot{E}^{(1)}(k + 1) = [E^{(0)}(1) - \frac{b_q}{a_q}e^{-ak} + \frac{b_q}{a_q}, \quad k = 1, 2, \ldots, N$$  \hspace{1cm} (24)

$$\dot{y}^{(1)}(k + 1) = \dot{y}^{(1)}(k + 1) + \dot{E}^{(1)}(k + 1), \quad k = 1, 2, \ldots, N$$  \hspace{1cm} (25)

$$\dot{y}^{(0)}(k) = \dot{y}^{(1)}(k) - \dot{y}^{(1)}(k - 1), \quad k = 2, 3, \ldots, N + 1$$  \hspace{1cm} (26)

where \( \dot{y}^{(0)}(k) \) is the predicted value of CPU utilization at the next time step. Li et al. [8] and Liu et al. [16] proposed a strategy to migrate VMs before a host is overloaded and reduce SLAVs, but we take another approach. When both the host CPU utilization and predicted utilization levels are higher than the high threshold, the VMs at the host need to migrate; otherwise, migration will not occur. This method can reduce the impact of load fluctuations on the migration accuracy.

To further reduce the performance degradation caused by VM migration, we use a dynamic threshold adjustment mechanism to optimize the consolidation timing. After a host is determined to be overloaded based on the high threshold and gray prediction, if the load fluctuation is small for the host, there is a small probability of an SLA violation occurring. In this case, the high threshold will be increased, and the migration threshold of the host will be changed to \( [T_{low}, T_{high} + T] \). At this time, the system can reduce the number of VM migrations when the load is relatively stable. To ensure the performance of the host and the VMs, the adjusted high threshold should not be higher than 0.9. We use the MAD to indicate the load fluctuations for the host over a period of time. By assuming \( U_i = [U_i(1), U_i(2), \ldots, U_i(n)] \) is the historical CPU utilization set, the MAD can be defined as follows:

$$MAD(U_i) = median_k(|U_i(k) - median_j(U_i(j))|)$$  \hspace{1cm} (27)

Set the load fluctuation limit as S. If MAD(i) is less than S, increase the high threshold by T; otherwise, the threshold will not change.

**Algorithm 1 Host Overload Detection**

**Input:** HostList

**Output:** OverUtilizedHostList

1. OverUtilizedHostList ← NULL;
2. for each host in HostList do
3.  if MAD(UiUtilizationHistory(host)) < S then
4.    T ← T_{high} + T;
5.  else T ← T_{high} + T;
6. end if
7. if utilization(host) > T & & Gray(utility)
8.  (host) > T then
9.  OverUtilizedHostList.add(host);
10. end if
11. end for
12. return OverUtilizedHostList

Algorithm 1 and Algorithm 2 describe the specific process of load detection. In Algorithm 1, a reasonable high threshold is determined based on the historical fluctuations in the load (lines 2-6). We use a gray prediction algorithm to determine whether a host is overloaded. If both the current load rate and the predicted load rate are higher than the high threshold, the host is determined to be overloaded (lines 7-10). In Algorithm 2, a list of hosts to be detected is created and all overloaded hosts, sleeping hosts, and hosts in migration are removed from the list (lines 2-5). When selecting an underloaded host, the host with the lowest utilization rate will be selected at each time step (lines 7-13).
migratable VMs, the selection index is calculated according to formula (28), and the VM with the smallest calculated index is selected for migration (lines 6-12). After each step in the VM selection process, it must be determined whether host overloading is still occurring after migrating VMs. If the host is overloaded, the VM selection process continues; otherwise, VM selection ends (lines 13-17).

1) TIME COMPLEXITY ANALYSIS
The total time complexity is $O$ (the predicted value of CPU utilization) + $O$ (the load fluctuation). Suppose the number of hosts in the datacenter is $m$; $O$ (calculated as the predicted value of CPU utilization) is $O(m)$, and $O$ (calculated as load fluctuation) is $O(m)$. Therefore, the total time complexity is $O(m)$.

B. VM SELECTION
For overloaded hosts, some VMs must be migrated until the CPU utilization level of the host is lower than the high threshold. A previous paper [15] proposed an MMT algorithm to migrate VMs. This algorithm selects the VM with the smallest $T_m$, to migrate each time. We consider CPU utilization based on the MMT algorithm; thus, at each step, VM migration reduces the CPU utilization of the host as much as possible, and the migration time can be reduced. According to the proposed algorithm, VMs with low migration times and high MIPS values will be the most likely to be selected. $V(h_i)$ is set for all the VMs on $host_i$, and the proposed selection algorithm is defined as follows:

$$
\forall v \in V(h_i) \text{, min } \left\{ \frac{RAM_v(v)}{BW_i} \times (1-U_{CPU_i}(v) \times \frac{CPU_v(v)}{CPU_i}) \right\} \tag{28}
$$

where $RAM_v(v)$ is the memory of $VM_v$, $BW_i$ is the available network bandwidth of $host_i$, $U_{CPU_i}(v)$ is the CPU utilization of $VM_v$, $CPU_v(v)$ is the MIPS allocated to the $VM_v$, and $CPU_i$ is the MIPS of $host_i$.

For an underloaded host, all VMs on the host need to migrate. When all VMs on an underloaded host migrate, the host is switched to a sleep state to save energy. The setting of the lower threshold can prevent the excessively aggressive consolidation of VMs, which can lead to frequent VM migrations and serious SLAVs.

Algorithm 3 describes the choice of VM. First, for all migratable VMs, the selection index is calculated according to formula (28), and the VM with the smallest calculated index is selected for migration (lines 6-12). After each step in the VM selection process, it must be determined whether host overloading is still occurring after migrating VMs. If the host is overloaded, the VM selection process continues; otherwise, VM selection ends (lines 13-17).

1) TIME COMPLEXITY ANALYSIS
The total time complexity is $O$ (the selection index of each + $O$ (selecting the VM with the best index). Suppose that the number of overloaded hosts in the datacenter is $n$ and the number of VMs for overloaded host $i$ is $m_i$; in this case, $O$(the selection index of each VM) is $O(\sum_{i=1}^{n} m_i)$, and $O$ (the load fluctuation) is $O(\sum_{i=1}^{n} m_i)$. Therefore, the total time complexity is $O(\sum_{i=1}^{n} m_i)$.

C. VM PLACEMENT
For overloaded and underloaded hosts, all VMs to be migrated should be allocated to new hosts. First, the selected VMs from the overloaded hosts are migrated, and then the VMs from the underloaded hosts are migrated. Wang and Tianfield [6] and Liu et al. [17] considered only the CPU utilization when placing VMs, and the performance of VMs was assessed based on multiple parameters. Therefore, considering multiple resources when integrating VMs can improve the rationality of the consolidation strategy. We consider CPU, memory, network bandwidth and storage resources, so the placement of VMs can be regarded as a multidimensional
bin-packing problem. If the VMs at a datacenter can be evenly
distributed among hosts, the probability of SLAVs occurring
at a host will be reduced. To ensure that the VMs can balance
the load of the datacenter and rationally use resources after
the consolidation of VMs, this paper proposes a balanced
placement strategy for VMs based on the discrete particle
swarm algorithm.

The strategy we propose uses the absolute error of the mean
to assess the load balance for the datacenter:

\[
\text{MEAD}(U_R) = \text{mean}(\{|U_R(i)| - \text{mean}(U_R(j))|\})
\]  

(29)

where \(U_R\) represents the utilization of a certain resource
by each host and \(\text{MEAD}(U_R)\) represents the mean absolute
deviation of \(U_R\). The objective function of the VM placement
strategy is as follows:

\[
F = \text{MEAD}(U_{\text{CPU}}) + \text{MEAD}(U_{\text{RAM}})
+ \text{MEAD}(U_{\text{BW}}) + \text{MEAD}(U_{\text{STORAGE}})
\]  

(30)

The goal of the placement strategy is to minimize the
value of the objective function. According to the objective
function, we propose an improved discrete particle swarm
optimization (DPSO) method.

Particle swarm optimization is a biology-based meta-
heuristic evolutionary computational method. The algorithm
is derived from studies of bird predation behavior, and the
basic objective is to find the optimal solution through col-
aboration and information sharing among individuals in a
group [21]. Particle swarm optimization simulates the birds
in a swarm as massless particles. These particles have only
two properties: speed and position. Each particle separately
searches for the optimal solution in the search space and
records it as the current individual optimal solution. The
individual optimal solution is shared with other particles in
the particle swarm to find the best individual optimal solution
as the current global optimal solution. Each particle in the
swarm adjusts its speed and position according to the current
individual optimal solution and the current global optimal
solution [43].

When DPSO is used to solve the global optimization prob-
lem, multiple feasible solutions for VM placement are com-
bined, and each feasible solution is a particle. Each particle
in the population has a position vector and a velocity vector [44].
The variables associated with each dimension of a particle are
continuously updated in an iterative process until the global
optimal position is determined. The position and velocity of
each particle are defined as follows:

\[
X_i = [x_{i,1}, x_{i,2}, \ldots, x_{i,n}]
\]  

(31)

\[
V_i = [v_{i,1}, v_{i,2}, \ldots, v_{i,n}]
\]  

(32)

where \(X_i\) is the position vector of particle \(i\), \(x_{i,1}\) is the ordinal
of the host where the first VM for particle \(i\) is placed, \(V_i\) is the
velocity vector of particle \(i\), and the elements in the vector are
the particle speeds for all dimensions.

In the population initialization phase, \(N\) feasible VM place-
ment schemes should be randomly generated. To guide the
particles toward the globally optimal position, this paper pro-
poses a resource-aware VM placement strategy. The scheme
obtained by this strategy is used as one of the particles in the
initial population.

The resource-aware VM placement strategy is based on the
remaining available resources for each host. The definitions
of the resource and host selection strategies are as follows:

\[
R_{\text{ava}}(h_i) = \frac{\text{CPU}_{\text{ava}}}{\text{MAX}_{\text{CPU}}} + \frac{\text{RAM}_{\text{ava}}}{\text{MAX}_{\text{RAM}}} + \frac{\text{BW}_{\text{ava}}}{\text{MAX}_{\text{BW}}} + \frac{\text{STORAGE}_{\text{ava}}}{\text{MAX}_{\text{STORAGE}}}
\]  

(33)

\[
h_{\text{target}} \in \text{HL} \forall h_i \in \text{HL}, R_{\text{ava}}(h_{\text{target}}) \geq R_{\text{ava}}(h_i)
\]  

(34)

where \(R_{\text{ava}}(h_i)\) is the definition of the remaining avail-
able resources for host \(h_i\); \(\text{CPU}_{\text{ava}}, \text{RAM}_{\text{ava}}, \text{BW}_{\text{ava}}\) and
\(\text{STORAGE}_{\text{ava}}\) are the remaining available MIPS, memory
capacity, network bandwidth and storage capacity resources
for \(h_i\), respectively; and \(\text{MAX}_{\text{CPU}}, \text{MAX}_{\text{RAM}}, \text{MAX}_{\text{BW}}\) and
\(\text{MAX}_{\text{STORAGE}}\) are the maximum MIPS, memory, network
bandwidth and storage capacity values for all hosts in the
datacenter, respectively. The resource-aware VM place-
ment strategy involves migrating VMs to the host with the
largest amount of remaining resources at a given
time.

In the particle search process, the optimal position obtained
by particle \(i\) is called the individual optimal solution of particle
\(i\), denoted as \(\text{Pbest}_i = [p_{1,i}, p_{2,i}, \ldots, p_{n,i}]\); the global
optimal position found by all particles in the population
is called the global optimal solution, denoted as \(\text{Gbest} =
[g_1, g_2, \ldots, g_n]\). The fitness value for each particle is cal-
culated according to the objective function, and the indi-
vidual optimal solution and the global optimal solution are
updated at each iteration. The particles continuously update
their speed and position according to the individual optimal
solution and the global optimal solution, and the updating
method is as follows [45]:

\[
v_{i,d}^{t+1} = \omega v_{i,d}^t + c_1 \times \text{rand}() \times (p_{i,d}^t - x_{i,d}^t) + c_2 \times \text{rand}() \times (g_{d}^t - x_{i,d}^t)
\]  

(35)

\[
x_{i,d}^{t+1} = x_{i,d}^{t} + \text{round}(v_{i,d}^{t+1})
\]  

(36)

\[-v_{\text{max}} \leq v_{i,d} \leq v_{\text{max}}
\]  

(37)

\[m_1 \leq x_{i,d} \leq m_n
\]  

(38)

where \(v_{i,d}^t\) is the velocity of particle \(i\) in the \(d\)-th dimension
at the \(t\)-th iteration, \(x_{i,d}^t\) is the position of particle \(i\) in the \(d\)-th
dimension at the \(t\)-th iteration, \(p_{i,d}^t\) is the individual optimal
solution of particle \(i\) in the \(d\)-th dimension at the \(t\)-th iteration,
\(g_{d}^t\) is the global optimal solution in the \(d\)-th dimension at
the \(t\)-th iteration, \(\omega\) is the inertia factor, \(c_1\) and \(c_2\) are learning
factors, \(\text{rand}()\) is a random number between 0 and 1, \(v_{\text{max}}\) is
the maximum velocity of particles, which is taken as 20% of
the number of hosts in this paper, \(m_1\) and \(m_n\) are the minimum
ordinal and maximum ordinal of the host, respectively, and
\(\text{round}(v_{i,d}^{t+1})\) represents the rounding of \(v_{i,d}^{t+1}\). Because the
The inertia factor and the learning factor are dynamically adjusted in a linear decreasing manner:

\[
\omega^i = (\omega_{ini} - \omega_{end}) \times (T - t)/T + \omega_{end}
\]

\[
c_i^1 = (c_{1,ini} - c_{1,end}) \times (T - t)/T + c_{1,end}
\]

\[
c_i^2 = (c_{2,ini} - c_{2,end}) \times (T - t)/T + c_{2,end}
\]

where \(\omega_{ini}\) and \(\omega_{end}\) are the maximum and minimum values of \(\omega\), respectively; \(c_{1,ini}\) and \(c_{1,end}\) are the maximum and minimum values of \(c_1\), respectively; \(c_{2,ini}\) and \(c_{2,end}\) are the maximum and minimum values of \(c_2\), respectively; \(t\) is the current number of iterations; and \(T\) is the maximum number of iterations.

To improve the global search ability of the particles in the early part of the search process, avoid falling into local optima, increase the local search ability in the late part of the search process, and accelerate the convergence speed, the inertia factor and the learning factor are dynamically adjusted in a linear decreasing manner:

\[
\omega^i = (\omega_{ini} - \omega_{end}) \times (T - t)/T + \omega_{end}
\]

\[
c_i^1 = (c_{1,ini} - c_{1,end}) \times (T - t)/T + c_{1,end}
\]

\[
c_i^2 = (c_{2,ini} - c_{2,end}) \times (T - t)/T + c_{2,end}
\]

where \(\omega_{ini}\) and \(\omega_{end}\) are the maximum and minimum values of \(\omega\), respectively; \(c_{1,ini}\) and \(c_{1,end}\) are the maximum and minimum values of \(c_1\), respectively; \(c_{2,ini}\) and \(c_{2,end}\) are the maximum and minimum values of \(c_2\), respectively; \(t\) is the current number of iterations; and \(T\) is the maximum number of iterations.

To increase the diversity of particles, we propose a particle mutation strategy. If the fitness value of particle \(i\) is less than the average fitness value of the particle swarm, map the particle speed to the interval \([0,1]\); then, mutate the particles with a certain probability. The specific implementation is as follows:

\[
s(v_{i,d}) = \frac{1}{1 + \exp(-v_{i,d})} \quad (42)
\]

\[
M = (M_{ini} - M_{end}) \times (T - t)/T + M_{end} \quad (43)
\]

\[
x_{i,d} = \begin{cases} \text{random}(m_0, m_n), & \text{rand}() < s(v_{i,d}) \times M \\ x_{i,d}, & \text{rand}() \geq s(v_{i,d}) \times M \end{cases} \quad (44)
\]

\(s(v_{i,d})\) represents mapping \(v_{i,d}\) to the interval \([0,1]\); \(M\) is the mutation probability of the particles; \(M_{ini}\) and \(M_{end}\) are the maximum mutation probability and the minimum mutation probability, respectively; and \(\text{random}(m_0, m_n)\) is a random number between the host numbers. Particle mutation is more likely to occur in the early part of the iteration process than in the later part, which is helpful for expanding the solution space of particles and avoiding local convergence; as the number of iterations increases, the probability of mutation will gradually decrease, making the particles converge quickly.

Algorithm 4 describes the specific process of the particle swarm consolidation algorithm. First, initialize the particle swarm parameters and the VM placement plan, as suggested in (34) (lines 1-2). By calculating the fitness function, determine the individual optimal position and the global optimal position of the initial particle swarm (lines 3-9). In the algorithm process, the velocity and position of each particle at each iteration are updated through (35) and (36), a new VM placement plan is generated, and the corresponding fitness function value is calculated (lines 11-21). For a plan with a fitness value lower than the average value, the placement plan needs to be updated through a mutation operation to ensure the diversity of the placement plan (lines 23-28). Finally, according to the results of the iteration, the individual optimal position and the local optimal position are calculated, and the optimal placement plan for the VM is obtained (lines 29-43).

**Algorithm 4 MDPSO VM Placement**

**Input:** HostList, vmsToMigrate

**Output:** MigrationMap

1. Initialize algorithm parameters
2. Initialize the population based on HostList and vmsToMigrate, and use eq. (34) as one of the initial particles
3. Calculate \(F\) based on eq. (30)
4. \(\text{gbest}.F \leftarrow \text{MAX}_\text{VALUE}\)
5. for each \(p \in \text{pop}\) do
   6. \(\text{pbest} \leftarrow X_0\)
   7. if \(X_0.F < \text{gbest}.F\) then
      8. \(\text{gbest} \leftarrow X_0\)
     9. end if
   10. while \(t < T\) do
      11. for each \(p \in \text{pop}\) do
         12. \(V_t \leftarrow \text{eq. (34)}\);
         13. \(X_t \leftarrow X_{t-1} + \text{rand}(V_t)\);
         14. for each \(x_i \in X_t\) do
            15. if \(x_i > m_n\) then
               16. \(x_i \leftarrow m_n\);
            17. end if
            18. if \(x_i < m_0\) then
               19. \(x_i \leftarrow m_0\);
            20. end if
         21. Calculate \(F*, \vec{F}\);
      22. end for
      23. for each \(p \in \text{pop}\) do
         24. if \(X_t.F < \vec{F}\) then
            25. \(X_t \leftarrow \text{eq. (44)}\);
         26. end if
         27. Calculate \(F\);
      28. end for
      29. for each \(p \in \text{pop}\) do
         30. if \(X_t.F < \text{Pbest}.F\) then
            31. \(\text{Pbest} \leftarrow X_t\);
         32. end if
      33. if \(X_t.F < \text{Gbest}.F\) then
         34. \(\text{Gbest} \leftarrow X_t\);
      35. end if
      36. end for
      37. \(t ++\);
   38. end while
   39. for each \(vm\) in vmsToMigrate do
      40. \(\text{allocatedHost} \leftarrow \text{Map}(vm, \text{allocatedHost})\);
   41. MigrationMap \(\leftarrow \text{Map}(\text{vm}, \text{allocatedHost})\);
   42. end for
43. return MigrationMap
1) TIME COMPLEXITY ANALYSIS
The total time complexity can be defined as $O$ (initializing the particle swarm) + $O$ (calculating the fitness function value of each particle) + $O$ (calculating the individual optimal solution) + $O$ (calculating the individual optimal position) + $O$ (calculating the global optimal solution) + $O$ (calculating the global optimal position) + $O$ (updating the inertia factor and learning factor) + $O$ (performing mutation operations).

Suppose that the number of VMs to be migrated at the datacenter is $m$, the number of particles is $N_p$, the number of iterations is $T$, and the number of particles that require mutation operations at the $i$th iteration is $N_{mi}$. In this case, $O$ (initializing the particle swarm) is $O(N_p \times m)$, $O$ (calculating the fitness function value of each particle) is $O(N_p \times T)$, $O$ (calculating the individual optimal solution) is $O(N_p \times T)$, $O$ (calculating the global optimal solution) is $O(N_p \times T)$, $O$ (calculating the global optimal position) is $O(N_p \times T)$, $O$ (updating the inertia factor and learning factor) is $O(T)$, and $O$ (mutation operations) is $O(\sum_{i=1}^{T} N_{mi} \times m)$. Therefore, the total time complexity is $O(N_p \times T \times m)$.

V. PERFORMANCE EVALUATION

A. EXPERIMENTAL SETTING
This article uses CloudSim as the simulation platform to test the performance of the proposed strategy. We simulated a datacenter consisting of 800 heterogeneous physical hosts in the experiment, including 400 HP ProLiant ML110 G4 servers (Intel Xeon 3040, 2 cores * 1.86 GHz) and 400 HP ProLiant ML110 G5 servers (Intel Xeon 3075, 2 cores * 2.26 GHz). There were four types of VMs in the experiment: high-CPU medium-instance (2500 MIPS, 0.85 GB), extra-large-instance (2000 MIPS, 3.75 GB), small-instance (1000 MIPS, 1.7 GB), and micro-instance (500 MIPS, 613 MB) VMs. To test the performance of the algorithm in an actual environment, the workload data used in the simulations were provided by the PlanetLab project [15]. PlanetLab is a computer cluster project distributed around the world. The project collects CPU utilization data for VMs from servers in more than 500 locations around the world. The project measures utilization every five minutes, and each measurement period is one day. We selected the workload traces for 10 days from March to April 2011. The specific data are shown in Table 1.

The parameters of the DPSO method used in this paper are shown in Table 2.

B. PERFORMANCE METRICS

1) ENERGY CONSUMPTION
Reducing the energy consumption of cloud datacenters is one of the goals of the VM dynamic consolidation algorithm proposed in this paper. According to the energy consumption modeling in Chapter III, the energy consumption of the host is closely related to CPU utilization. Table 2 shows the relationship between energy consumption by the test servers and CPU utilization.

2) NUMBER OF VM MIGRATIONS
VM migration requires a certain amount of network bandwidth resources and has a negative impact on the performance of VMs during the migration process. Excessive migrations can lead to an increase in SLAVs. Therefore, the number of VM migrations should be reduced to ensure the service quality of the datacenter.

3) SLAV TIME PER ACTIVE HOST (SLATAH)
SLATAH represents the percentage of time that active hosts experience a CPU utilization level of 100% [15]. In this case, the application performance of each VM is limited by the capacity of the host, and the VM cannot meet the required performance level. The definition of SLATAH is as follows:

$$SLATAH = \frac{1}{N} \sum_{i=1}^{N} \frac{T_{SLAV_i}}{T_{active_i}}$$

where $N$ is the number of hosts; $T_{SLAV_i}$ is the total time that $host_i$ has experienced a utilization level of 100%, resulting in SLAVs; and $T_{active_i}$ is the total active time for $host_i$. 

---

| Date     | VMs | Mean   | St. dev. |
|----------|-----|--------|----------|
| 2011/03/03 | 1052 | 12.31% | 17.09%   |
| 2011/03/06 | 898  | 11.44% | 16.83%   |
| 2011/03/09 | 1061 | 10.70% | 15.57%   |
| 2011/03/22 | 1516 | 9.26%  | 12.78%   |
| 2011/03/25 | 1078 | 10.56% | 14.14%   |
| 2011/04/03 | 1463 | 12.39% | 16.55%   |
| 2011/04/09 | 1358 | 11.12% | 15.09%   |
| 2011/04/11 | 1233 | 11.56% | 15.07%   |
| 2011/04/12 | 1054 | 11.54% | 15.15%   |
| 2011/04/20 | 1033 | 10.43% | 15.21%   |

| Parameter | Value |
|-----------|-------|
| Number of particles $N_p$ | 50 |
| Number of iterations $T$ | 100 |
| $\omega_w$ | 0.9 |
| $\omega_{max}$ | 0.4 |
| $c_{1,rand}$ | 3.5 |
| $c_{2,rand}$ | 1 |
| $M_{w}$ | 0.25 |
| $M_{max}$ | 0.05 |
TABLE 3. Power consumption at different levels of utilization in Watts.

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

TABLE 4. Average results using PlanetLab trace with several metrics.

| Strategy    | Energy (kWh) | VM migrations | SLA (×10⁻³) | SLATAH (%) | PDM (%) | ESV (×10⁻³) |
|-------------|--------------|---------------|--------------|------------|---------|-------------|
| DVFS        | 829.50       | 0             | 0            | 0          | 0       | 0           |
| IQR-MC-1.5  | 177.02       | 23880.7       | 6.879        | 6.925      | 0.099   | 1.206       |
| IQR-MMT-1.5 | 187.53       | 26496.9       | 3.288        | 5.023      | 0.065   | 0.607       |
| LR-MC-1.2   | 146.84       | 23540.4       | 7.641        | 7.469      | 0.103   | 1.111       |
| LR-MMT-1.2  | 161.87       | 27632.0       | 4.974        | 6.213      | 0.080   | 0.795       |
| LRR-MC-1.2  | 146.84       | 23540.4       | 7.641        | 7.469      | 0.103   | 1.111       |
| LRR-MMT-1.2 | 161.87       | 27632.0       | 4.974        | 6.213      | 0.080   | 0.795       |
| MAD-MC-2.5  | 172.76       | 23143.2       | 7.022        | 7.019      | 0.101   | 1.204       |
| MAD-MMT-2.5 | 183.491      | 26505.1       | 3.348        | 5.054      | 0.065   | 0.606       |
| THR-MC-0.85 | 172.345      | 22984.0       | 7.092        | 7.019      | 0.100   | 1.208       |
| THR-MMT-0.85| 181.57       | 25620.4       | 3.242        | 4.993      | 0.064   | 0.581       |
| DVC-EQ      | 122.766      | 2374.7        | 0.189        | 2.344      | 0.010   | 0.021       |

4) PERFORMANCE DEGRADATION DUE TO MIGRATION (PDM)

PDM is the overall performance of degradation for VMs due to migration [15]. The definition of PDM is as follows:

\[ PDM = \frac{1}{M} \sum_{i=1}^{M} \frac{C_{\text{deg}}}{C_{\text{cap}}} \]  \hspace{1cm} (46)

where \( M \) is the number of VMs; \( C_{\text{deg}} \) is the estimated value of the performance degradation of VM\(_i\) due to migrations, accounting for approximately 10% of the CPU utilization; and \( C_{\text{cap}} \) is the total CPU capacity requested by the VM\(_i\) during its active time.

5) SLAVs

SLAVs for users occur due to VM consolidation. SLAVs are related to the SLATAH and PDM, and they reflect a service quality degradation caused by host overload and VM migration. The definition of a SLAV is as follows:

\[ SLAV = SLATAH \times PDM \]  \hspace{1cm} (47)

6) ESV

ESV is the product of energy consumption and SLAVs, and it is a comprehensive performance metric for the VM consolidation strategy. The definition of ESV is as follows:

\[ ESV = E \times SLAV \]  \hspace{1cm} (48)

C. SIMULATION RESULTS AND ANALYSIS

During the experiment, we select several other VM consolidation algorithms for experimental comparison. These algorithms include eight traditional ones: Iqr-Mc-1.5, Iqr-Mm-1.5, Lr-Mc-1.2, Lr-Mmt-1.2, Lrr-Mc-1.2, Lrr-Mmt-1.2, Mad-Mc-2.5, Mad-Mmt-2.5, Thr-Mc-0.85, Thr-Mmt-0.85, two biological heuristic algorithms based on the ant colony system: ACS-VM [14] and AVVMC [13], as well as the DVFS strategy. Table 4 shows the average of all metrics obtained after using 10 sets of data for all strategies.

According to the data in Table 4, VM consolidation is vastly superior to DVFS based on energy savings, and the proposed GM-DPSO strategy is better than other dynamic consolidation strategies for all metrics. In terms of energy consumption, compared with other dynamic consolidation strategies, GM-DPSO can reduce energy consumption by up to 34.53%. In terms of service quality, the GM-DPSO algorithm reduces SLATAH and PDM by up to 68.62% and 90.29%, and the overall SLAV and ESV values are reduced by 97.53% and 98.26%, respectively. ACS-VM is superior to AVVMC in terms of performance. Compared with ACS-VM, GM-DPSO can reduce energy consumption, the VM migration number, SLAVs and ESV by 7.61%, 89.9%, 94.17% and 94.74%, respectively. According to the average experimental results, GM-DPSO achieves the best performance in terms of energy consumption and service quality, and provides the most optimal service quality.
To specifically analyze the performance of the dynamic consolidation algorithm under different load conditions, we select “20110306” data (number of VMs: 898; average utilization: 11.44%) as a low-load case and select “20110403” data (number of VMs: 1463; average utilization: 12.93%) as a high-load case.

Figure 1 shows the performance of various dynamic VM migration strategies in terms of energy consumption. Under low and high loads, GM-DPSO can reduce energy consumption by up to 34.43% and 32.36%, respectively. In addition to GM-DPSO, LR-MC and LRR-MC obtain the best performance in terms of energy consumption (20.03% higher than GM-DPSO under low loads; 17.64% higher than GM-DPSO under high loads). The Iqr-Mmt-1.5 strategy has the highest energy consumption, and the methods that use the MMT strategy in the VM selection algorithm are inferior to those that use the MC algorithm based on energy consumption because the MMT algorithm only considers the migration time when selecting VMs and does not consider the impact of the selected VMs on reducing energy consumption. MC considers the relations among VMs and can quickly reduce host CPU utilization. Compared with the ACS-VM and AVVMC algorithms, GM-DPSO can reduce energy consumption by 8.68% and 14.28%, respectively, under low-load conditions. Under high-load conditions, the three algorithms perform similarly. Both ACS-VM and AVVMC place VMs on as few hosts as possible to reduce energy consumption. GM-DPSO can significantly reduce the number of VMs to be migrated (as shown in Figure 2) and the energy consumption of VMs. Dynamic threshold adjustment helps hosts with stable loads to accommodate additional VMs. The GM-DPSO algorithm effectively reduces the number of running hosts and reasonably allocates resources among hosts, thereby reducing the number of times a host is switched. Therefore, the algorithm has certain advantages in reducing energy consumption.

As shown in Figure 2, GM-DPSO can greatly reduce the number of VM migrations. According to the test results, the average number of migrations for all benchmark algorithms is 26567.05, and the average number of migrations for GM-DPSO is only 2303. GM-DPSO introduces gray prediction and dynamic threshold adjustment for overload detection, thus effectively avoiding invalid migrations. The placement strategy based on load balancing reduces the possibility of a host being overloaded and decreases the number of subsequent migrations. In addition, this article reduces the lower threshold in low-load detection to prevent excessive VM consolidation.

Figures 3, 4, and 5 show the performance of different consolidation algorithms in terms of SLAs. These figures show that GM-DPSO can effectively avoid excessive consolidation by considering the balance of the load distribution, thus reducing the probability of resource shortages for some nodes. GM-DPSO can reduce the overall SLAV value by up to 97.67% and 98.48% under low and high loads, respectively; SLATAH caused by host overload can be reduced by up to 68.77% and 75.58%; and PDM caused by VM migrations can be reduced by up to 90.91% and 90%. Compared with other traditional algorithms, ACS-VM and AVVMC are superior based on SLAV, but the performance of GM-DPSO is
generally superior. In addition, the two ant colony consolidation algorithms focus on reducing energy consumption at the datacenter, and the consolidation strategy adopted does not significantly reduce the number of VMs to be migrated; thus, these methods are not as good as GM-DPSO in terms of SLAV.

SLATAH reflects the degradation of service quality due to host overload. The LR and LRR algorithms have high SLATAH metrics under different loads, but they perform well in terms of energy consumption. The LR and LRR
algorithms encourage the consolidation of VMs, but they also increase the possibility of host overload. The MMT algorithm is superior to the MC algorithm based on SLATAH because the MMT algorithm reduces the time for VM migrations and excessive CPU use during the migration process, thus decreasing the host overload rate. We propose a VM placement strategy based on improved DPSO to consolidate VMs. This approach fully considers load balancing when placing VMs, thus ensuring that the remaining resource space for each host is balanced and preventing the overload of hosts caused by the centralized placement of VMs. In addition, the lower threshold limits the number of VMs to be migrated, which reduces the high load rate of a host due to the excessive consolidation of VMs.

The PDM is closely related to the number of VM migrations. According to the analysis results, gray prediction and dynamic threshold adjustment can reduce the migration of VMs to decrease the PDM. In addition, the VM placement strategy of GM-DPSO not only reduces the host overload rate but also decreases VM migration caused by overload and improves PDM performance. The VM selection strategy proposed in this paper considers the migration time and CPU utilization, so it can synchronously optimize SLATAH and PDM. Based on the above analysis, SLATAH and PDM are closely interlinked. A SLATAH value that is too high increases the number of overloaded hosts and further increases the number of VMs to be migrated. The migration of underloaded hosts can reduce the number of servers operating at a datacenter, and the migration of VMs increases the load rate of active hosts and the probability of host overload.

ESV reflects the overall levels of energy consumption and service quality, as shown in Figure 6. Under different load conditions, GM-DPSO is superior to all other strategies in terms of the ESV result. Thus, GM-DPSO can effectively guarantee the service quality of a datacenter while reducing energy consumption. The proposed strategy provides high-performance cloud computing and balances the remaining resource space for each computing node, thereby improving the scalability of the system and reducing the operating cost. Therefore, a win-win situation is achieved for users and suppliers.

VI. CONCLUSION
In this article, we propose a dynamic VM consolidation (GM-DPSO) strategy based on a gray model and an improved discrete particle swarm algorithm. First, we use the gray prediction algorithm and dynamic threshold adjustment mechanism to determine the timing of migration. Then, we select the VMs to be migrated according to the migration times and CPU utilization levels and complete the consolidation with a VM placement strategy based on the improved discrete particle swarm algorithm. We analyzed the factors that affect the quality of system service and developed a corresponding combination strategy. We use actual load tests to assess the performance of the consolidation strategy and compare GM-DPSO with multiple algorithms. The experimental results show that the GM-DPSO algorithm can effectively reduce the energy consumption of the system and improve the quality of service under different load conditions. The overload detection strategy based on gray prediction and dynamic threshold adjustment can effectively reduce the migration of VMs and SLAs. The DPSO VM placement strategy searches for the optimal solution in the global scope. During the search process, the algorithm dynamically adjusts various parameters and introduces a mutation factor to increase the richness of the population and avoid local optima. The DPSO strategy provides a reasonable allocation method for datacenter resources, reduces the probability of service quality degradation, and improves the service stability of the system. Overall, the combined algorithm adopted by GM-DPSO greatly reduces the energy consumption of VM migration and effectively decreases the number of hosts running at a datacenter, thereby reducing the energy consumption of the entire system.

At an actual datacenter, to ensure the stability and security of the operation, backups are required for each host. The host backup can transfer data to the backup host when the host experiences sudden downtime to continue to serve users. Therefore, in future work, we hope to consider sufficient backup redundancy in the consolidation algorithm so that it can meet the needs of practical work and be used at actual datacenters.

REFERENCES
[1] L. Sharififi, L. Cerdá-Alabern, F. Freitag, and L. Veiga, “Energy efficient cloud service provisioning: Keeping datacenter granularity in perspective,” J. Grid Comput., vol. 14, no. 2, pp. 299–325, Jun. 2016.
[2] D. K. Kang, F. Alhazemi, and S. H. Kim, “Dynamic VM consolidation for energy efficient cloud datacenters,” in Proc. 6th Interface Conf. Cloud Comput., 2015, pp. 70–80.
[3] M. H. Ferdaus, M. Murshed, and Z. Mahmood, “Energy-aware VM consolidation in IaaS cloud computing,” in Proc. Cloud Comput., 2014, pp. 179–208.
[4] A. Beloglazov, J. Abawajy, and R. Buyya, “Energy-aware resource allocation heuristics for efficient management of datacenters for cloud computing,” Future Generat. Comput. Syst., vol. 28, no. 5, pp. 755–768, 2012.
[5] M. C. S. Filho, C. C. Monteiro, P. R. M. Inácio, and M. M. Freire, “Approaches for optimizing VM placement and migration in cloud environments: A survey,” J. Parallel Distrib. Comput., vol. 111, pp. 222–250, Jan. 2018.
[6] H. Wang and H. Tianfield, “Energy-aware dynamic vm consolidation for cloud datacenters,” IEEE Access, vol. 6, pp. 15259–15273, 2018.
[7] A. Ammar, J. Luo, D. Zao, and Z. Tang, “Intra-balance VM placement for effective reduction in energy consumption and SLA violation,” IEEE Access, vol. 7, pp. 72387–72402, 2019.
[8] L. Li, J. Dong, D. Zuo, and J. Wu, “Sla-aware and energy-efficient VM consolidation in cloud datacenters using robust linear regression prediction model,” IEEE Access, vol. 7, pp. 9490–9500, 2019.
[9] S. Homsi, S. Liu, G. A. Chaparro-Baquer, O. Bai, S. Ren, and G. Quan, “Workload consolidation for cloud datacenters with guaranteed QoS using request reneging,” IEEE Trans. Parallel Distrib. Syst., vol. 28, no. 7, pp. 2103–2116, Jul. 2017.
[10] B. Gul, I. A. Khan, S. Mustafa, O. Khalid, S. S. Hussain, D. Daneyce, and A. R. Nawaz, “CPU and RAM energy-based SLA-aware workload consolidation techniques for clouds,” IEEE Access, vol. 8, pp. 62990–63003, 2020.
[11] M.-H. Malekloo, N. Kara, and M. El Barachi, “An energy efficient and SLA compliant approach for resource allocation and consolidation in cloud computing environments,” Sustain. Comput., Inf. Syst., vol. 17, pp. 9–24, Mar. 2018.
[12] Z. Li, X. Yu, L. Yu, S. Guo, and V. Chang, “Energy-efficient and quality-aware VM consolidation method,” *Future Gener. Comput. Syst.*, vol. 102, pp. 789–809, Jan. 2020.

[13] M. H. Ferdaus, M. Murshed, R. N. Calheiros, and R. Buyya, “VM consolidation in cloud datacenters using ACO metaheuristic,” in *Euro-Par Parallel Processing*. Berlin, Germany: Springer, 2014, pp. 306–317.

[14] F. Farahmakan, A. Ashraf, T. Pahikkala, P. Liljeberg, J. Plosila, I. Porres, and H. Tenhunen, “Using ant colony system to consolidate VMs for green cloud computing,” *IEEE Trans. Services Comput.*, vol. 8, no. 2, pp. 187–198, Mar./Apr. 2015.

[15] A. Beloglazov and R. Buyya, “Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of VMs in cloud datacenters,” *Concurrency Comput., Pract. Exper.*, vol. 24, no. 13, pp. 1397–1420, 2012.

[16] Y. Liu, X. Sun, W. Wei, and W. Jing, “Enhancing energy-efficient and QoS dynamic VM consolidation method in cloud environment,” *IEEE Access*, vol. 6, pp. 31224–31235, 2018.

[17] F. Liu, Z. Ma, B. Wang, and W. Lin, “A VM consolidation algorithm based on ant colony system and extreme learning machine for cloud datacenter,” *IEEE Access*, vol. 8, pp. 53–67, 2020.

[18] M. Gamal, R. Rizk, H. Mahdi, and B. E. Elagraghi, “Osmotic bio-inspired load balancing algorithm in cloud computing,” *IEEE Access*, vol. 7, pp. 42735–42744, 2019.

[19] C. Gao, Z. Hu, Z. Xiong, and Q. Su, “Gray prediction evolution algorithm based on accelerated even gray model,” *IEEE Access*, vol. 8, pp. 107941–107957, 2020.

[20] Z. Zhou, Z. Hu, J.-Y. Yu, J. Abawajy, and M. Chowdhury, “Energy-efficient VM consolidation algorithm in cloud datacenters,” *J. Central South Univ.*, vol. 24, no. 10, pp. 2331–2341, Oct. 2017.

[21] A. Ibrahim, M. Noshy, H. A. Ali, and M. Badawy, “PAPSO: A power-aware VM placement technique based on particle swarm optimization,” *IEEE Access*, vol. 8, pp. 81747–81764, 2020.

[22] S. Y. Z. Fard, M. R. Ahmadi, and S. Adabi, “A dynamic VM consolidation technique for QoS and energy consumption environment,” *J. Supercomput.*, vol. 73, no. 10, pp. 4347–4368, Oct. 2017.

[23] R. Yadav, W. Zhang, O. Kaivartya, P. R. Singh, I. A. Elgendi, and Y.-C. Tian, “Adaptive energy-aware algorithms for minimizing energy consumption and SLA violation in cloud computing,” *IEEE Access*, vol. 6, pp. 55923–55936, 2018.

[24] N. T. Hieu, M. D. Francesco, and A. Yla-Jaaski, “VM consolidation with multiple usage prediction for energy-efficient cloud datacenters,” *IEEE Trans. Services Comput.*, vol. 13, no. 1, pp. 186–199, Jan./Feb. 2020.

[25] Z. Li, C. Yan, X. Yu, and N. Yu, “Bayesian network-based VMs consolidation method,” *Future Gener. Comput. Syst.*, vol. 69, pp. 75–87, Apr. 2017.

[26] M. Mishra and A. Sahoo, “On theory of VM placement: Anomalies in existing methodologies and their mitigation using a novel vector allocation scheme for heterogeneous cloud datacenters,” *J. Netw. Comput. Appl.*, vol. 15, no. 6, pp. 832–847, Dec. 2011.

[27] D. Feng, Z. Wu, D. Zuo, and Z. Zhang, “A multiobjective migration algorithm as a resource consolidation strategy in cloud computing,” *PLOS ONE*, vol. 14, no. 2, Feb. 2019, Art. no. e0211729.
YONG PAN received the B.S. degree from Tian-gong University. He currently works as the IT Manager at CCSSR. His current research interests include cloud computing, big data, and artificial intelligence.

YI ZHOU received the B.S. degree from Jinan University. He currently works as the General Manager with MXSun. His current research interests include cloud computing, big data, and artificial intelligence.

ZHAO ZHOU is currently pursuing the bachelor’s degree with the School of Electronical Information Science and Technology, Kunming University of Science and Technology. His current research interests include microcomputer principles and computer network principles.

* * *