IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received June 7, 2020, accepted June 18, 2020, date of publication June 24, 2020, date of current version July 7, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3004615

Container Migration Mechanism for Load Balancing in Edge Network Under Power Internet of Things

ZITONG MA, SUJIE SHAO, (Member, IEEE), SHAOYONG GUO, ZHILI WANG, FENG QI, AND AO XIONG, (Member, IEEE)

State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China

Corresponding authors: Sujie Shao (buptssj@bupt.edu.cn) and Ao Xiong (xiongao@bupt.edu.cn)

This work was supported in part by the Beijing Natural Science Foundation through the Research on Adaptive Fault Recovery Mechanism for Electric Power IoT under Grant 4194085, and in part by the Fundamental Research Funds for the Central Universities under Grant 2019RC08.

ABSTRACT As a novel computing technology closer to business ends, edge computing has become an effective solution for delay sensitive business of power Internet of Things (IoT). However, the uneven spatial and temporal distribution of business requests in edge network leads to a significant difference in business busyness between edge nodes. Due to the natural lightweight and portability, container migration has become a critical technology for load balancing, thereby optimizing the resource utilization of edge nodes. To this end, this paper proposes a container migration-based decision-making (CMDM) mechanism in power IoT. First, a load differentiation matrix model between edge nodes is constructed to determine the timing of container migration. Then, a container migration model of load balancing joint migration cost (LBJC) is established to minimize the impact of container migration while balancing the load of edge network. Finally, the migration priority of containers is determined from the perspective of resource correlation and business relevance, and by introducing a pseudo-random ratio rule and combining the local pheromone evaporation with global pheromone update at the same time, a migration algorithm based on modified Ant Colony System (MACS) is designed to utilize the LBJC model and thus guiding the choice of possible migration actions. The simulation results show that compared with genetic algorithm (GA) and Space Aware Best Fit Decreasing (SABFD) algorithm, the comprehensive performance of CMDM in load balancing joint migration cost is improved by 7.3% and 12.5% respectively.

INDEX TERMS Container migration, load balancing, migration cost, edge computing, power Internet of Things.

I. INTRODUCTION

With the development of smart grid construction and the rising popularity of connected devices and sensors in smart cities, massive transmission and processing data has led to a significant increase in the network transmission pressure and computing load of cloud center under the gradually formed new business requirements of large connection, wide coverage and intelligence in power IoT, which makes it difficult to meet the business processing delays [1]. By deploying intelligent processing equipment or servers with capabilities of communication access, business processing and data storage closer to the edge of user terminals, namely the edge node to provide services, edge computing provides an effective solution for alleviating the pressure of computing and storage in cloud center and ensuring the delay demand of business of power IoT [2]. As shown in Figure 1, the terminal devices communicate with the edge nodes through wired, WiFi, or 4G/5G, and send the task requests to edge nodes instead of the cloud platform, which reduces the network transmission time and increases the terminals' intelligence capabilities at the same time.

However, under the framework of edge computing, the power IoT still faces a series of problems. The uneven spatial and temporal distribution of service requests in edge network with limited resources of the edge node leads to a significant difference in the busyness of business between edge nodes. As a result, some of the edge nodes are too

The associate editor coordinating the review of this manuscript and approving it for publication was Taehong Kim.

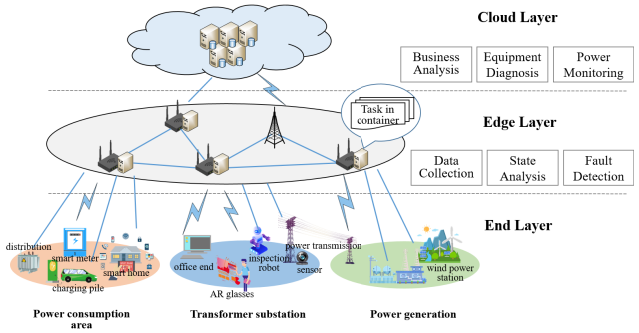

FIGURE 1. The power IoT structure based on edge computing.

busy to timely meet the business's requirement of connection and processing, while resources of the others are not fully utilized. Therefore, it is necessary to implement load balancing of the edge network, thereby optimizing the resource utilization of the edge nodes. By migrating a running virtual machine (VM) from one physical machine to another without disconnecting the applications, real-time migration of virtual machines has become an effective technology for balancing the load between edge nodes. However, the slow startup speed of virtual machine makes it difficult to meet the Quality of Service (QoS) requirements of delay sensitive business of power IoT. At the same time, as the number of virtual machines deployed in the edge network increases, the performance of virtual machines will decrease significantly.

To overcome the known deficiencies of VM, a light-weight virtualization technology called container has been widely used. Compared with VMs, containers use hierarchical storage that separates the mirror layer from the data layer, and are isolated by control groups (Cgroups) rather than hypervisors, thus enabling rapid startup, deployment, and release [3]. In addition, container is lightweight and supports migration for it is an operating system-level virtualization technology that shares the kernel of host's operating system [4]. Therefore, it is necessary to formulate a reasonable container migration strategy to realize load balancing of the edge network and minimize the impact of container migration on users at the same time on the premise of meeting the requirements of multiple types of business of power IoT.

To solve the above problem, on the one hand, a reasonable migration timing and container migration selection strategy need to be established, and on the other hand, container migration algorithms require rational selection. Concerning to the problem of migration timing, some existing solutions typically rely on static thresholds while the others use prediction techniques such as linear regression [5] and machine learning [6]. However, when only taking into consideration the state of host, the overall condition of edge network may be ignored. And since the container migration (CM) problem is NP-hard [7], some researchers put forward heuristic algorithms to get approximate optimal solutions, but it is easy to fall into the local optimal. Bio-inspired meta-heuristic algorithms, such as Ant Colony System (ACS) algorithm, Genetic Algorithm and Particle Swarm Optimization (PSO) algorithm can not only avoid falling into a locally optimal solution but also get high-quality approximation [8]. However, some meta-heuristic algorithms such as GA and PSO require special encoding for combinatorial optimization problems because they are designed for continuous problems originally. In comparison, ACS algorithm designed for discrete problems is viewed as an effective approach to solve the CM problem. For solving the multi-objective optimization-scheduling problem in cloud computing, [9] proposed an improved Ant Colony Optimization (ACO) algorithm, which considers not only the utilization of computing and storage resources of physical nodes but also the number of micro-service requests and failure rate. In [10], a multi-objective Ant Colony algorithm was proposed for finding the optimal mapping of VMs on the servers of a data center by searching for Pareto-optimal solutions based on two dimensions, namely the power consumption and resource wastage. However, the above methods are specific to container redeployment problem and lack the consideration of migration cost and delay in edge computing.

In summary, in order to address the problem of container migration for load balancing of edge network in power IoT, we propose a container migration-based decision-making mechanism and a migration algorithm based on improved ACS is designed for model optimization. The specific contributions of this paper are summarized as follows:

- The container migration problem for load balancing of edge network in power IoT is described as a multi-objective combination optimization problem under QoS constraints. By considering the balancing of resource utilization and remaining resource, network transmission delay and container migration downtime, a container migration model of load balancing joint migration costs is established to minimize the impact of container migration while balancing the load of edge network.
- The selection strategy of containers to be migrated is considered from the perspective of resource correlation and business relevance in order to ensure the delay demand of business of power IoT while further reducing the migration cost.
- A modified ACS migration algorithm based on CMDM mechanism is designed to solve the container migration problem. By introducing a pseudo-random ratio rule and combining the local pheromone evaporation with global pheromone update at the same time, the algorithm's convergence speed is ensured while the exploration ability is enhanced.

The rest of this paper is organized as follows: Section II reviews the related work. Section III presents the container migration-based decision-making mechanism, and a migration algorithm based on improved discrete ACS for solving container migration problem is described in section IV. Section V shows the performance of proposed algorithm by simulation, and the last chapter summarizes the full text and draws conclusions.

II. RELATED WORK

A. LOAD BALANCING IN EDGE COMPUTING

Load balancing for edge network is a process of redistributing load among edge nodes to improve resource utilization and reduce response time [11]. At the same time, ensuring the load balancing of the edge network can cope with the emergency situation in power IoT, and has a good bearing capacity for the peak of business requests. At present, there are two main types of load balancing in distributed environment, namely the static load balancing and dynamic load balancing [12]. By expressing the problem of load balancing under the edge computing architecture as an optimization problem, reference [13] proposed a scalable algorithm to find the redirection of tasks among a given set of edge nodes in the network, thereby minimizing the average response time. In [14], a cloud-assisted framework of mobile edge computing was proposed to enhance the computing capabilities of edge network, and a workload scheduling mechanism was proposed to balance the system latency and cost at the same time. In order to balance the resource utilization of large data centers, reference [15] studied the container placement and redistribution problems in the industrial environment, and proposed a worst fit decreasing algorithm and a two-stage algorithm respectively for optimizing the given initial distribution of containers by migrating containers among servers.

In short, most of the current solutions for load imbalances under edge computing scenarios are to formulate static load balancing strategies, that is, to implement the initial task allocation or resource allocation based on the load balancing model, which makes it difficult to deal with dynamic problems under the edge network.

B. CONTAINER MIGRATION

Container migration is considered as an efficient and lightweight virtualization technology for dynamic load balancing. At present, the two main research directions of container migration are migration strategies and concrete migration implementation. Reference [16] proposed a method to minimize the service delay in a scenario with two edge servers, in which the computation delay was reduced by migration and the communication delay was improved by transmission power control at the same time. In [17], a container scheduling algorithm based on ACO was proposed to balance the resource utilization and improve the application performance. Considering the power consumption and delay, reference [3] modeled the container migration strategy as multiple dimensional Markov Decision Process (MDP) spaces and proposed a deep reinforcement learning algorithm to achieve fast decision-making. A VM dynamic migration framework for load balancing was designed in [18]. By dividing the server into four different load states, a dynamic migration strategy of VM was formulated to guide the migration decision. In [19], a cost model of migration time was constructed and two dynamic migration strategies for different application scenarios were proposed, namely the load balancing and fault tolerance. To flexibly meet users' demands in cloud computing, reference [20] proposed a grey relational analysis (GRA) and technique for order preference and a two-level hybrid heuristic algorithm was designed to consolidate resources in order to reduce costs and energy consumption.

From the existing work above, we can see that there is just a little research on container migration for load balancing in edge network. In addition, the existing VM-based migration mechanism for load balancing in cloud computing fails to consider the migration cost, which makes it difficult to meet the QoS requirements of delay sensitive business of power IoT in edge computing.

III. CONTAINER MIGRATION-BASED DECISION-MAKING MECHANISM

In this section, we propose a CMDM mechanism for making container migration decision whereby the migration model is built to make a tradeoff between the load balance and migration cost. The scheme is described in three parts in Figure 2. First, we measure the load diversity of different edge nodes and decide whether to perform migration. Then, we propose a container migration model based on load balancing and migration cost. Finally, we provide a container migration strategy, which including the selection of migration containers and immigration edge nodes. Specific processes of the three parts are shown as below.

A. LOAD BALANCING DETECTION

A large number of tasks generated by the terminal devices are offloaded through the access network to the corresponding containers within the edge node (EN) closer to the terminal for execution. A container can be abstracted as an ''active'' application environment that has acquired or occupied a portion of the edge node's resources.

Assume that the set of ENs in the edge network is $\mathcal{N} = \{n_1, n_2, \cdots, n_J\}$, where J is the number of ENs and the set of containers is $\mathcal{C} = \{c_1, c_2, \cdots, c_I\}$, where I is the number of containers. Each EN has multiple types of resources including CPU, memory, storage and countless others, so the collection of resources can be expressed as $\mathcal{R} = \{r_1, r_2, \cdots, r_K\}$. For each $n_j \in \mathcal{N}$, let $\overrightarrow{W_j} = (W_j^1, W_j^2, \cdots, W_j^K)$ denote its capacity of all types of resources, where W_j^k represents the maximum amount of $k-th$ resource that n_j can provide, and let $\overrightarrow{u_j} = (u_j^1, u_j^2, \cdots, u_j^K)$ denote its resource utilization. The maximum data transmission rate between n_j and $n_{j'}$ is defined as $Band_{j,j'}$. For each $c_i \in \mathcal{C}$, let $w_i = [\overrightarrow{d_i}, t_i^{sta}, t_i^{del}, Tol_i]$ denote its operating information, where $\overrightarrow{d_i} = (d_i^1, d_i^2, \cdots, d_i^K)$ represents the requirement for all types of resources, t_i^{sta} represents the startup time of c_i, t_i^{del} represents the computing delay when the resource requirement are met, and Tol_i represents the delay threshold within toleration. The resource requirement of containers considered in this paper are the number of virtual resource units after standardization.

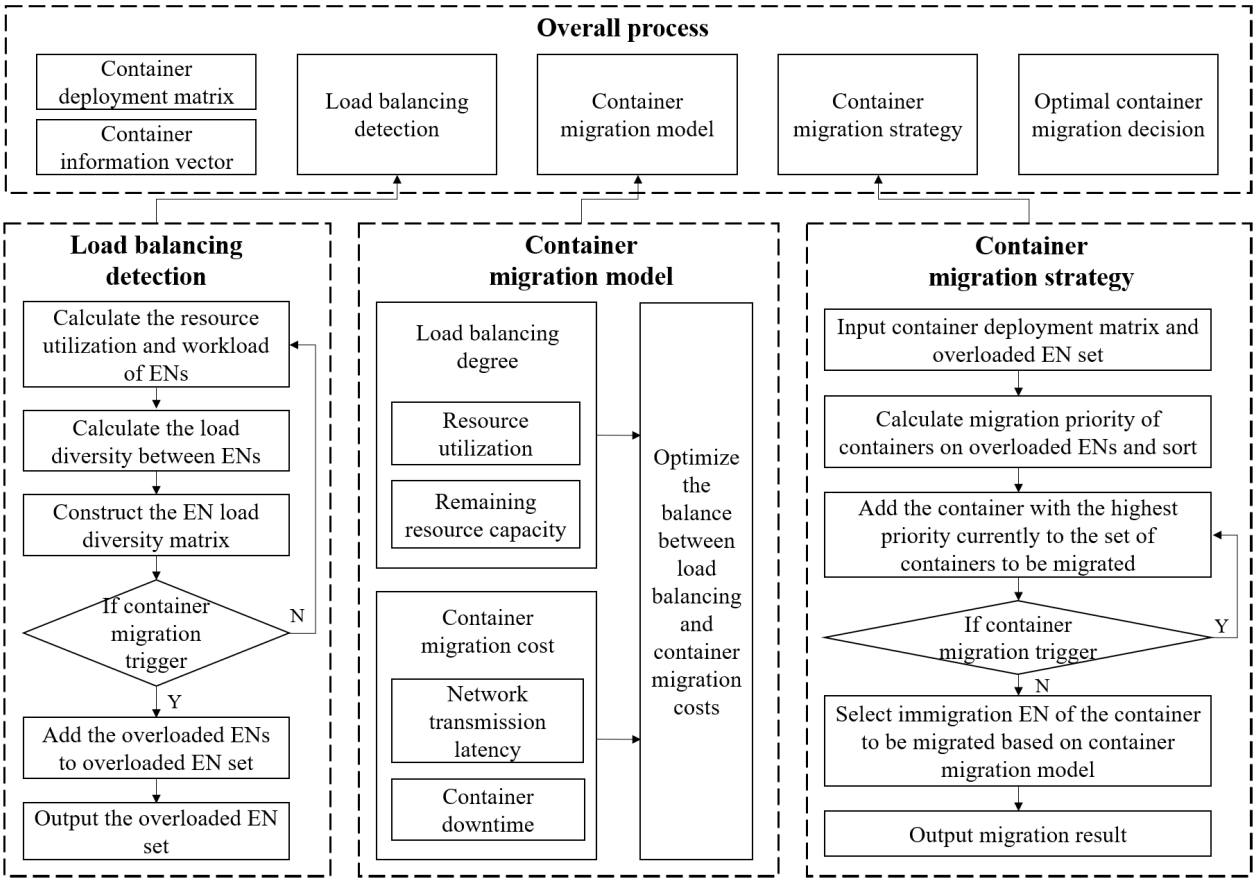

FIGURE 2. The framework of CMDM mechanism.

Taking a binary indicator $x_{i,j}$ to donate the container placement decision variable. Let $x_{i,j} = 1$ if c_i is placed at n_j, and $x_{i,j} = 0$ else. The set of containers deployed on n_j is defined as $V(n_j)$.

The load Γ_{n_j} for each n_j is calculated using the following equation.

$$\Gamma_{n_j} = \sum_{r_k \in \mathcal{R}} \eta_k u_j^k \tag{1}$$

where η_k represents the weight of r_k on the load of n_j, and u_j^k is the utilization of r_k on n_j. The calculation formula is as follows.

$$u_j^k = \frac{\sum\limits_{c_i \in \mathcal{C}} x_{i,j} d_i^k}{W_j^k} \tag{2}$$

Hence, the load diversity between n_j and $n_{j'}$ is:

$$h_{n_j,n_{j'}} = \frac{\Gamma_{n_j}}{\Gamma_{n_{j'}}} \tag{3}$$

Derived from the above, the EN load diversity matrix $H_{J\times J}$ can be represented as follows.

$$H_{J\times J} = \begin{bmatrix} 1 & h_{n_1,n_2} & \cdots & h_{n_1,n_J} \\ h_{n_2,n_1} & 1 & \cdots & h_{n_2,n_J} \\ \vdots & \vdots & & \vdots \\ h_{n_J,n_1} & h_{n_J,n_2} & \cdots & 1 \end{bmatrix} \tag{4}$$

Two aspects are considered concerning to the load imbalance of edge network, namely the load diversity between any two ENs and the utilization of each type of resources of a single EN. By calculating the load diversity, frequent container migration between ENs can be avoided, and the resource fragmentation problems caused by over-utilization of certain types of resources can be averted by taking into consideration the resource utilization of ENs at the same time.

Therefore, given the threshold σ of load diversity and the upper threshold Thr_k of resource utilization, the containers migration-triggered metric due to load imbalance of the edge network is:

$$\begin{aligned} &\forall n_j \in \mathcal{N}, \quad \exists u_j^k \geq Thr_k \\ &\forall n_j \in \mathcal{N}, \quad r_k \in \mathcal{R}, \exists h_{n_j,n_{j'}} \geq \sigma \end{aligned} \tag{5}$$

B. LBJC CONTAINER MIGRATION MODEL

First, container migration and replacement are defined as two binary variables: $y_{j,j'}^i$ and $x_{i,j}$. If c_i is migrated from n_j to $n_{j'}$, that is $x_{i,j} = 1$ and $\widetilde{x_{i,j'}} = 1$, $y_{j,j'}^i = 1$, otherwise $y_{j,j'}^i = 0$.

The migration cost is mainly formed by two components, namely the network latency caused by the transmission of intermediate and final data between two edge nodes and the migration time of the container itself, that is the downtime. The downtime is mainly affected by the memory size of containers, and the larger the memory, the longer

TABLE 1. Definition of variables used in the article.

Symbol	Description
$\mathcal{N}$	the set of EN
$\mathcal{C}$	the set of container
$\mathcal{R}$	the set of resource
$V(n_j)$	the set of container deployed on EN j
W_j^k	the maximum amount of resource k that EN j can provide
u_j^k	the utilization of resource k on EN j
$\widetilde{u_j^k}$	the utilization of resource k on EN j after migration
$\overline{\widetilde{u_j^k}}$	the average utilization of resource k on EN j after migration
d_i^k	the requirement for resource k of container i
S_j^k	the residual available resources of resource k on EN j
$Band_{j,j'}$	the maximum data transmission rate between EN j and j'
t_i^{sta}	the startup time of container i
t_i^{del}	the computing delay of business in container i if the resource requirement is met
t_i^{alr}	the execution time of container i
Tol_i	the delay threshold within toleration of container i
$x_{i,j}$	whether container i is deployed on EN j
$\widetilde{x_{i,j}}$	whether container i is deployed on EN j after migration
$y_{j,j'}^i$	whether container i is migrated from EN j to EN j'
Λ_i	the amount of data transmitted by container i during the process of migration
Γ_{n_j}	the load of EN j
$h_{n_j,n_{j'}}$	the load diversity between EN j and EN j'
$H_{J\times J}$	the load diversity matrix of ENs
$m_{i,j}^{MMT}$	the migration probability of container i on EN j based on migration time
$m_{i,j}^{MM}$	the migration probability of container i on EN j based on migration frequency
$m_{i,j}^{TT}$	the migration probability of container i on EN j based on the amount of migration data

the downtime [21]. Compared with network latency and downtime, the migration decision delay is negligible [22]. So, when a container c_i is migrated from EN n_j to $n_{j'}$, the migration cost Mig_{cost}^i can be defined as follows.

$$Mig_{cost}^i = \frac{\Lambda_i + d_i^{k_{mem}} / \sum_{c_i \in \mathcal{C}} x_{i,j} d_i^{k_{mem}}}{Band_{j,j'}} \tag{6}$$

where Λ_i is the amount of data transmitted by c_i during the migration process, and $d_i^{k_{mem}}$ is the memory resource allocated to it. Therefore, the total migration cost can be calculated as follows.

$$Mig_{cost}^{total} = \sum_{c_i \in \mathcal{C}} y_{j,j'}^i Mig_{cost}^i \tag{7}$$

After the migration of containers to be migrated, the load balancing degree of the edge network $Load_{balance}$ is measured from two aspects, namely the balance of resource utilization between ENs of the same kind and the equilibrium degree of residual resource of different types of resources on the ENs.

$$Load_{balance} = U_{balance}^{total} + V_{balance}^{total} \tag{8}$$

where $U_{balance}^{total}$ is the balance of resource utilization and $V_{balance}^{total}$ is the equilibrium degree of residual resources, that are defined as follows.

$$U_{balance}^{total} = \sum_{r_k \in \mathcal{R}} \sum_{n_j \in \mathcal{J}} \frac{\left(\widetilde{u_j^k} - \overline{\widetilde{u_j^k}}\right)^2}{J} \tag{9}$$

$$V_{balance}^{total} = \sum_{n_j \in \mathcal{J}} \sum_{r_{k_1}, r_{k_2} \in \mathcal{R}} max\left\{0, \frac{S_j^{k_1}}{W_j^{k_1}} - \frac{S_j^{k_2}}{W_j^{k_2}}\right\} \tag{10}$$

where $\widetilde{u_j^k}$ denotes the utilization of r_k on n_j, $\overline{\widetilde{u_j^k}}$ is the mean utilization of r_k of all ENs and S_j^k refers to the residual available resource r_k on n_j, which can be calculated as follows.

$$S_j^k = W_j^k - \sum_{c_i \in \mathcal{C}} \widetilde{x_{i,j}} d_i^k \tag{11}$$

From the above, we can see that the smaller the value of balancing of resource utilization joint residual resource, the higher the load balancing degree of edge network.

Finally, the objective of container migration is defined as a weighted sum of all the factors defined above.

$$F = \theta(U_{balance}^{total} + V_{balance}^{total}) + \gamma Mig_{cost}^{total} \tag{12}$$

where θ and γ are the weights of load balancing degree of edge network and container migration cost, and $\theta + \gamma = 1$.

To minimize the above objective, containers should be migrated to the most suitable ENs, but this process should satisfy some strict constraints.

$$\begin{aligned}
&C1: \sum_{c_i \in \mathcal{C}} x_{i,j} d_i^k \le W_j^k, \quad \forall n_j \in \mathcal{N}, r_k \in \mathcal{R} \\
&C2: t_i^{del} + y_{j,j'}^i Mig_{cost}^i \le Tol_i, \quad \forall c_i \in \mathcal{C},\ n_j \in \mathcal{N} \\
&C3: \sum_{n_j \in \mathcal{N}} x_{i,j} = 1, \quad \forall c_i \in \mathcal{C} \\
&C4: \sum_{n_j \in \mathcal{N}} \widetilde{x_{i,j}} = 1, \quad \forall c_i \in \mathcal{C}
\end{aligned} \tag{13}$$

Constraint C1 imposes that the utilization of any type of resources on EN should be lower than the maximum capacity of that on EN. Constraint C2 indicates that the total delay of tasks in the container to be migrated should not exceed the delay threshold that the user can tolerate. Constraint C3 and C4 stipulate that a container can only be allocated to one EN for processing.

To sum up, in order to make a trade-off between the load balancing degree of edge network after migration and the migration cost of containers during migration, this paper describes the container migration problem for load balancing of the edge network in power IoT as a multi-objective optimization problem under QoS constraints and presents the migration model as follows.

$$\begin{aligned}
P1: \ &\min \left\{\theta(U_{balance}^{total} + V_{balance}^{total}) + \gamma Mig_{cost}^{total}\right\} \\
&s.t. \ \text{C1,C2,C3,C4} \\
&x_{i,j}, \widetilde{x_{i,j}}, y_{j,j'}^i \in \{0, 1\}, \quad \forall c_i \in \mathcal{C},\ n_j \in \mathcal{N}
\end{aligned} \tag{14}$$

C. MIGRATION STRATEGY

The purpose of migration strategy is to migrate some containers from overloaded ENs to relatively idle ENs with higher migration efficiency when container migration is triggered. In short, the two main issues to be solved by the migration strategy are the selection of containers to be migrated and container remapping.

1) MIGRATION CONTAINER SELECTION

Containers to be migrated from the overloaded EN are selected according to two aspects, namely the resource correlation and business relevance. Resource correlation prefers to migrate a container with relatively short migration time and more efficiency for eliminating overload. Business correlation takes into consideration the execution progress of the business hosted by the container.

The CPU and memory size of the container are direct factors affecting migration downtime, resource loss and energy consumption [23]. The smaller the utilization of CPU and memory, the shorter the container migration downtime, the lower the energy consumption of the ENs and the less the impact on performance. Thus, the migration probability $m_{i,j}^{MMT}$ of the container c_i on the EN n_j based on migration time can be expressed as:

$$m_{i,j}^{MMT} = \frac{\omega_{cpu} d_i^{k_{cpu}}}{\sum\limits_{c_i \in \mathcal{C}} x_{i,j} d_i^{k_{cpu}}} + \frac{\omega_{mem} d_i^{k_{mem}}}{\sum\limits_{c_i \in \mathcal{C}} x_{i,j} d_i^{k_{mem}}} \tag{15}$$

where ω_{cpu} and ω_{mem} are pre-specified weights of the impact of container utilization of CPU and memory on migration downtime, and $\omega_{cpu} + \omega_{mem} = 1$.

However, minimizing the migration time cannot improve the load state of edge network. In contrast, the total migration downtime may increase due to frequent container migration. Therefore, considering the single container migration time, the Euclidean distance between containers and ENs is calculated at the same time to reduce the migration frequency. Note that, the larger the distance, the effect of the EN is dominant.

$$\wp_{i,j} = \sum_{r_k \in \mathcal{R}} \frac{\varpi_k}{\sqrt{\left(u_j^k - u_i^k\right)^2}} \tag{16}$$

where ϖ_k is the weight of the influence of r_k on the load of n_j, and is defined as follows:

$$\varpi_k = \frac{u_j^k}{\sum\limits_{r_k \in \mathcal{R}} u_j^k}$$
$$\sum_{r_k \in \mathcal{R}} \varpi_k = 1 \tag{17}$$

Thus, the migration probability $m_{i,j}^{MM}$ of c_i on n_j based on migration frequency can be expressed as:

$$m_{i,j}^{MM} = \frac{\wp_{i,j}}{\sum\limits_{c_i \in \mathcal{C}} \wp_{i,j}} \tag{18}$$

From the perspective of the business hosted by the container, container migration will inevitably result in the network overhead of data transmission between two ENs. The amount of data transmitted is mainly affected by the progress of business execution, that is, the more the progress of business execution, the smaller the relative amount of data generated. Therefore, the migration probability $m_{i,j}^{TT}$ of c_i on n_j based on the amount of migration data can be expressed as.

$$m_{i,j}^{TT} = \frac{t_i^{\mathrm{al}r}}{t_i^{del}} \tag{19}$$

where $t_i^{\mathrm{al}r}$ is the execution time of c_i.

In summary, the migration priority of container c_i on the EN n_j is defined as:

$$m_{i,j} = -\lambda m_{i,j}^{MMT} + \mu m_{i,j}^{MM} + \psi m_{i,j}^{TT} \tag{20}$$

where λ, μ and ψ are the weights of migration time, migration frequency and the amount of migration data of c_i on n_j, and $\lambda + \mu + \psi = 1$. And according to Equation (20), we can get a set $M_j = \left\{m_{1,j}, m_{2,j}, \cdots m_{|V(n_j)|,j}\right\}$, which represents the migration priority between the EN n_j and containers developed on it.

2) IMMIGRATION EDGE NODE SELECTION

Guided by the migration model proposed in the previous section, n_j is selected as the immigration EN of the container to be migrated based on the following equation.

$$\begin{aligned} n_j = \ & \arg\min \left\{\theta(U_{balance}^{total} + V_{balance}^{total}) + \gamma Mig_{cost}^{total}\right\} \\ & s.t.\ \ \mathrm{C1,C2,C3,C4} \\ & x_{i,j}, \widetilde{x_{i,j}}, y_{j,j'}^{i} \in \{0,1\}, \quad \forall c_i \in \mathcal{C}, n_j \in \mathcal{N} \end{aligned} \tag{21}$$

IV. CMDM_MACS MIGRATION ALGORITHM

In order to obtain the global optimal container migration decision, an optimal ACS algorithm for discrete problems is designed based on CMDM mechanism. Compared with other meta-heuristic algorithms such as GA and PSO, ACS algorithm adopts pheromone strategy to make the experience information shared among different groups.

The ACS algorithm simulates the feeding process of ants to complete the scheduling of container migration, as shown in Figure 3. The algorithm is summarized as follows:

- Ant Ant_l is randomly placed to a container c_i to be migrated.
- Ant_l selects a mapping tuple $< c_i, n_j >$ with a certain probability $p_{i,j}$, that is deploying c_i to n_j according to the pheromones $\tau_{i,j}$ and heuristic information $\eta_{i,j}$. After that, c_i can be put into tabu list $Tabu_l$ of Ant_l.
- Ant_l returns to the next container in the set of containers to be migrated C_{mig}, and repeats the previous process to complete the following migration allocation, getting a migration plan.
- That all the ants complete the allocation of all the containers to be migrated once, can be regarded as

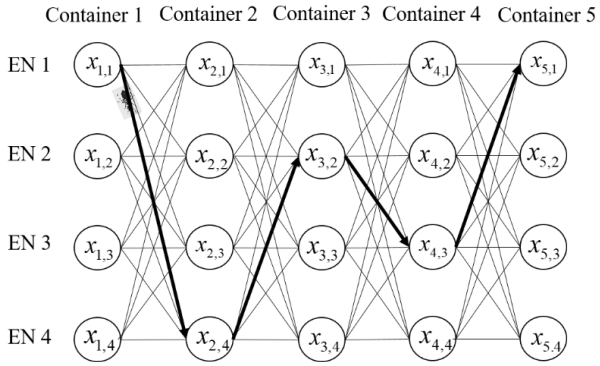

FIGURE 3. Diagram of ACS Algorithm based on container migration.

one iteration. The algorithm terminates until the maximum number of iterations is reached.

A. PHEROMONE UPDATING

In reality, the pheromone is a kind of chemical substance that ants use to communicate with each other. Ants find the source of food along the way by sensing the pheromones released by other ants [24]. In the process of container migration, the ACS algorithm saves the search experience of ants by creating the pheromone matrix $\left[\tau_{i,j}\right]_{N\times M}$, where $\tau_{i,j}$ represents the pheromone that migrated c_i to n_j and the larger the value of $\tau_{i,j}$, the more likely the ant is to choose tuple $< c_i, n_j >$ in the previous iteration process. The initial value of the pheromone is defined as follows:

$$\tau_0 = \frac{1}{\left|C_{mig}\right|} \tag{22}$$

where $\left|C_{mig}\right|$ is the number of containers to be migrated.

After selecting a new mapping relation tuple $< c_i, n_j >$, the ant updates the pheromone of this traversed mapping relation using the following local pheromone update rule.

$$\tau_{i,j} = (1-\rho_l)\,\tau_{i,j} \tag{23}$$

where $\rho_l \in [0, 1]$ is the local pheromone evaporating parameter and the larger the value of ρ_l, the less pheromone remains on $< c_i, n_j >$.

After all the ants complete building migration schemes, the quality of all current solutions is evaluated according to the objective function, and the best one is selected to perform the following global pheromone update rule to preserve the experience of the global optimal solution.

$$\begin{aligned} \tau_{i,j} &= \tau_{i,j} + \rho_g \Delta\tau \\ \Delta\tau &= \begin{cases} \dfrac{1}{F\left(X^+\right)}, & if\ \ \widetilde{x_{i,j}} = 1\ in\ X^+ \\ 0, & otherwise \end{cases} \end{aligned} \tag{24}$$

where $\rho_g \in [0, 1]$ is the global pheromone evaporation parameter, $\Delta\tau$ is the increment of additional pheromones and X^+ is the global optimal solution in an iteration.

B. HEURISTIC INFORMATION

Unlike the pheromone that provides historical information, heuristic information is better selected by greedy strategy in the current local situation. The heuristic factor is expressed as $\eta_{i,j}$, representing the desirability of migrating c_i to n_j. The heuristic factor reduces the blindness of ant search and is combined with the pheromone in ACS to construct the migration decision scheme.

Based on the CMDM mechanism proposed in this paper, heuristic information $\eta_{i,j}$ is mainly calculated based on the migration cost of migrating c_i to n_j and the remaining resources on n_j after migration.

For the migration cost incurred by migrating c_i to n_j, heuristic information $\eta^1_{i,j}$ is expressed as:

$$\eta^1_{i,j} = Band_{j',j} \tag{25}$$

For the impact of container migration on the target node n_j, heuristic information $\eta^2_{i,j}$ is designed to avoid resource overload while balancing the surplus of different types of resources, and is expressed as:

$$\eta^2_{i,j} = \frac{\sum\limits_{r_k\in\mathcal{R}} \frac{s^k_j - d^k_i}{w^k_j}}{\sum\limits_{\forall r_{k_1}, r_{k_2}\in\mathcal{R}} max\left\{0, \frac{s^{k_1}_j - d^{k_1}_i}{w^{k_1}_j} - \frac{s^{k_2}_j - d^{k_2}_i}{w^{k_2}_j}\right\}} \tag{26}$$

By synthesizing the above heuristic information of the two objectives, the expectation of Ant_l to migrate the container c_i to the EN n_j can be formulated as:

$$\eta_{i,j} = \eta^1_{i,j} \times \eta^2_{i,j} \tag{27}$$

C. PSEUDO-RANDOM-PROPORTION RULE

Ant_l tends to select the next mapping relation tuple with more pheromone and higher expectation from the current path. However, to avoid falling into local optimum, ants will select the tuples to traverse according to the following pseudo-random-proportion rule j.

$$j = \begin{cases} arg\ max_{n_u\in\Theta_l(i)}\left\{[\tau_{i,u}]^\alpha \times [\eta_{i,u}]^\beta\right\}, & if\ q \le q_0 \\ J, & otherwise \end{cases} \tag{28}$$

where q is a random number uniformly distributed in $[0, 1]$ and $q_0 \in [0, 1]$ is a predefined parameter. When $q \le q_0$, Ant_l selects n_j with the max product as the destination EN of c_i. Otherwise, the destination EN is selected based on the following roulette wheel rule.

$$p_{i,j} = \begin{cases} \dfrac{[\tau_{i,j}]^\alpha \times [\eta_{i,j}]^\beta}{\sum_{n_u\in\Theta_l(i)}[\tau_{i,u}]^\alpha \times [\eta_{i,u}]^\beta}, & if\ n_j \in \Theta_l(i) \\ 0, & otherwise \end{cases}, \quad q > q_0 \tag{29}$$

where $p_{i,j}$ represents the probability that c_i selects n_j as its destination EN. $\Theta_l(i)$ is a set of effective ENs of Ant_l that

Algorithm 1: Container Migration Based On Modified ACS Algorithm

Input: $\mathcal{N}'$, C_{mig}, nI, nA, nC, α, β, ρ_l, ρ_g, q_0, ε
Output: X

```
for p ∈ [1, nI] do
    for l ∈ [1, nA] do
        for each container c_i in C_mig do
            for each EN n_j in Θ_l(i) do
                calculate p_{i,j} according to Eq.(27);
            end
            generate a random variable q ∈ [0, 1];
            if q ≤ q_0 then
                choose an EN according to Eq.(28);
            else
                choose an EN according to Eq.(29);
            end
            add tuple < c_i, n_j > to X_l;
            update the local pheromone according to
             Eq.(23) and the resources;
            x_{i,j} = 1;
            put c_i into Tabu_l;
        end
        calculate the score of X_l according to Eq.(14);
    end
    X^+ ← argmax_{X_l ∈ X} {f(X_l)};
    update the global pheromone according to Eq.(24);
end
```

satisfy the constraints, and is defined as follows.

$$\Theta_l(i) = \left\{ n_j \;\middle|\; \sum_{c_s \in \mathcal{C}} x_{s,j} d_s^k + d_i^k \le Thr_k \times W_j^k,\ \forall r_k \in \mathcal{R} \right\} \tag{30}$$

V. EXPERIMENTAL RESULTS AND ANALYSIS

In this section, we evaluated the performance of CMDM_MACS in a heterogeneous edge network environment and compared it with other algorithms.

In the edge computing system, the physical environment is a square area with a side length of 5km. The number of ENs is 20, the number of containers is 30 $\sim$ 210, and the position of containers is randomly generated in this area. The relevant parameters in the experiment are listed in table 2 and are applicable to the simulation example unless otherwise specified. In addition, we have uploaded the detailed algorithm code to https://github.com/TTtong567/CMDM_MACS.

A. STRATEGY PERFORMANCE

In order to verify the feasibility and effectiveness of the CMDM mechanism, the following three container selection strategies are selected for comparison.

For minimizing the impact of VM migration on users, [25] proposed a minimum migration time (MMT) strategy, which selects a VM that takes the shortest possible time to complete the migration relatively to the other VMs allocated to the overloaded EN. Reference [26] proposed a high CPU utilization-based selection (HS) strategy that selects a VM making the highest CPU utilization in the overloaded host, thereby decreasing the workload of the host quickly and minimizing the number of potential migrations needed. Aiming at simplifying the process of container selection, [27] proposed a random selection (RS) strategy, which selects a container to be migrated according to a uniformly distributed discrete random variable.

In Figure 4, the migration cost under four container selection strategies mentioned above are compared. With the continuous increase of the number of containers, the resource occupancy rate of ENs goes up, and the load difference of edge network becomes more and more obvious, which will cause more containers migrating to relatively idle ENs to ensure the delay constraint of business. As a result, as the number of containers increases, the cost of migration in the system increases linearly. When the total number of containers is less than 60, the MMT strategy shows better performance. However, with the increasing load of some ENs, MMT strategy falls to improve the migration cost due to frequent container migration. On the contrary, the selection strategy proposed in this paper tends to migrate containers

TABLE 2. Simulation parameters.

Symbol	Description	Values
W_j^1	CPU number of EN j	32Cores
W_j^2	memory capacity of EN j	16-32GB
W_j^3	storage capacity of EN j	100-300GB
d_i^1	CPU resources required by container i	1-4Cores
d_i^2	memory resources required by container i	1-8GB
d_i^3	storage resources required by container i	2-16GB
$Band_{j,j'}$	maximum data transmission rate between EN j and j'	100-300MB/s
Λ_i	the amount of data transmitted by container i	100KB-1MB
Tol_i	delay threshold within toleration of container i	5-500ms
σ	threshold of load diversity	2
Thr_1	upper threshold of CPU resource utilization	0.6
Thr_2	upper threshold of memory resource utilization	0.9
Thr_3	upper threshold of storage resource utilization	0.9
ω_1, ω_2	weights of the impact of container utilization of CPU and memory on migration downtime	0.5
λ	weight of migration time of container i on EN j	$1/3$
μ	weight of migration frequency of container i on EN j	$1/3$
ψ	weight of the amount of migration data of container i on EN j	$1/3$
θ	weight of load balancing degree of edge network	$1/2$
γ	weight of container migration cost	$1/2$
nI	number of iterations	200
nA	number of ants	10
α	parameter of pheromone updating	1
β	parameter of heuristic information	2
ρ_l	parameter of local pheromone evaporating	0.1
ρ_g	parameter of global pheromone evaporation	0.8
q_0	fixed parameter that determines the relative importance of cumulative experience with random selection	0.1
nC	number of chromosomes	20
p_r	parameter of replication probability	0.2
p_c	parameter of crossover probability	0.8
p_m	parameter of mutation probability	0.2

TABLE 3. Experimental results comparison with different number of containers under different strategies.

Number of containers	CMDM strategy		MMT strategy		HS strategy		RS strategy	
	Load balabce	Migration cost	Load balance	Migration cost	Load balance	Migration cost	Load balance	Migration cost
60	3.75	5.10	3.88	5.10	3.85	6.18	4.81	7.00
90	4.69	6.35	4.81	6.75	4.70	7.25	4.79	8.04
120	4.83	7.80	4.98	8.05	5.19	8.15	5.20	8.18
150	4.25	8.05	4.31	8.15	4.38	9.02	4.39	8.51
180	6.19	9.11	6.22	9.25	6.26	11.10	6.24	11.01
210	5.58	10.93	5.83	12.18	5.86	12.20	5.72	12.21

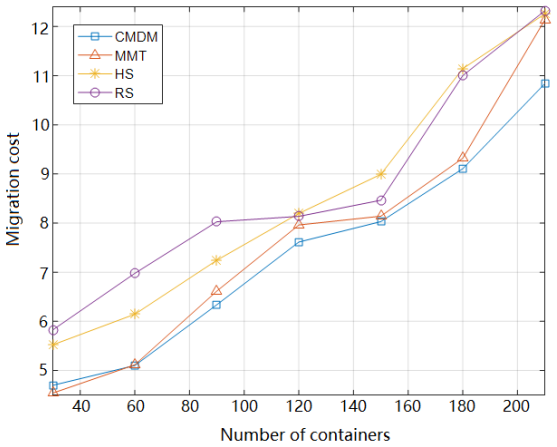

FIGURE 4. Cost of migration with different number of containers under different strategies.

that can significantly reduce the load while considering the reduction of the single migration time. Therefore, as the number of containers increases, the CMDM migration strategy shows better performance.

Figure 5 shows the comparison of load balancing degree of edge network under four migration selection strategies. The load balancing degree is measured from two aspects, namely the balance of resource utilization between ENs of the same kind and the equilibrium degree of residual resource of different types of resources on the ENs, and the smaller

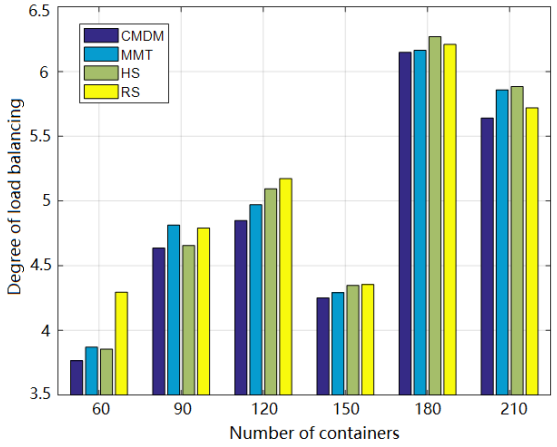

FIGURE 5. Degree of load balancing with different number of containers under different strategies.

the value of load balancing is, the more balanced the load is between ENs. As can be seen from the figure, as the number of containers continues to increase, the CMDM migration strategy not only improves the cost of system migration but also ensures the load balancing of edge network.

Table 3 summarizes the simulation results of load balancing degree and migration cost with different number of containers under different container migration selection strategies, which further verifies the feasibility and effectiveness of the CMDM mechanism that taking into consideration both the resource correlation and business relevance.

B. ALGORITHM PERFORMANCE

Based on the CMDM mechanism, the container migration algorithm based on improved ACS proposed in this paper is compared with the Elitist Ant System (EAS) algorithm, the genetic algorithm proposed in [28] and the Space Aware Best Fit Decreasing (SABFD) algorithm proposed in [29] to verify the performance from aspects of convergence, load before and after migration and optimization objective.

The EAS algorithm is the first improvement of the basic Ant System (AS) algorithm. It adds an enhancement method to the hitherto optimal path based on the pheromone update rule of the original. The genetic algorithm works as follows. First of all, set the number of iterations and chromosomes, and initialize the first generation population. Then, calculate the value of fitness function so as to obtain the natural selection probability. After that, individuals are selected, crossed, and mutated during the iterative phase. At last, the calculation is terminated by taking the individual with the maximum fitness obtained in the evolutionary process as the output of the optimal solution. The related parameters of the genetic algorithm are listed in Table 2. The SABFD algorithm is a kind of greedy algorithm of local optimum. The containers selected to migrate are sorted in a decreasing order of CPU utilization, and the EN with the minimum migration cost and the maximum load balancing degree of the edge network after the first container being migrated to will be selected as the target node of this container.

Although the comparison algorithms mentioned above can be implemented in practical scenarios, the time complexity and performance vary from each other significantly.

As shown in Algorithm 1, we can conclude that the time complexity of the MACS algorithm is $O(nI \cdot nA \cdot |C_{mig}| \cdot J)$, while the time complexity of GA and SABFD algorithm are

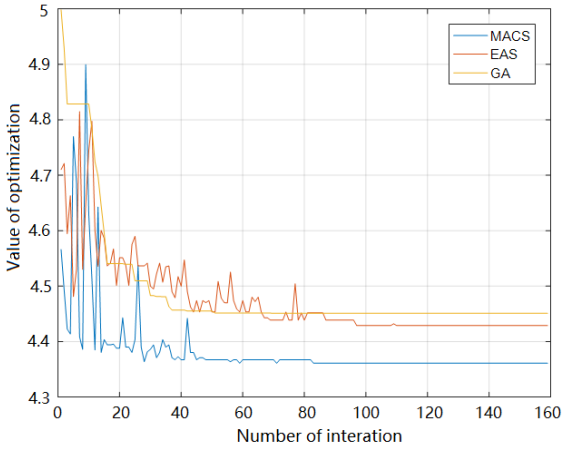

FIGURE 6. Relationship between optimization value and iteration times under different algorithms.

$O(nI \cdot nC \cdot |C_{mig}| \cdot J)$ and $O(|C_{mig}| \cdot \log^{|C_{mig}|} \cdot J)$ respectively, where nI is the number of iterations, nA is the number of ants, nC is the number of chromosomes, $|C_{mig}|$ is the number of containers to be migrated and J is the number of ENs.

Figure 6 shows the performance of the MACS, EAS and GA algorithms as the number of iterations increases during container migration. The smaller the value of optimization, the better the performance. It can be seen that the EAS algorithm has the advantage of better performance than the GA algorithm after convergence, but the convergence speed is slower. The MACS algorithm proposed in this paper improves the EAS algorithm by introducing pseudorandom ratio rule and combining local pheromone evaporation with global pheromone update, so as to provide a better solution by strengthening the connection between tuples under better migration results. Therefore, the MACS algorithm can overcome the disadvantages of convergence speed, and the final performance is better than the EAS and GA algorithms.

In Figure 7, the performance of MACS under different parameters of the pseudo-random ratio rule are compared as

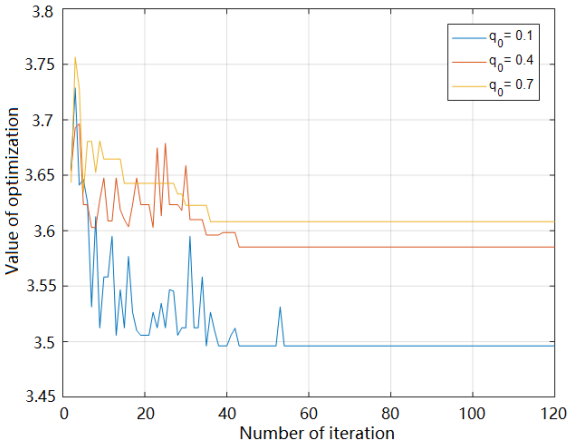

FIGURE 7. Relationship between optimization value and iteration times under different parameter.

the number of iterations increases during container migration. By changing the value of q_0, we can effectively adjust the balance between "development" and "exploration" of the ACS algorithm, so as to determine whether to focus on developing the area near the optimal path or explore the new area. It can be seen that the smaller the value of q_0 is, the more the algorithm tends to explore new regions so as to avoid falling into the local optimal solution, thereby obtaining better performance after convergence. On the contrary, the larger the value of q_0 is, the more the algorithm tends to develop the region near the optimal path with poor performance. Based on this, we choose the value of q_0 as 0.1 to obtain the minimum load balancing degree and migration cost.

Figure 8 shows the load changes of each EN before and after migration. It can be seen that the load of EN7 was over 80%, while that of EN1, EN2 and EN6 was less than 30% before the container migration, in which case the container virtualization layer of the heavily loaded EN will be congested, and the queuing delay will be greatly increased, resulting in the dissatisfaction of QoS requirements of some businesss, while the EN with less load cannot be fully utilized, due to the regularity and predictability of terminal requests in power IoT. After the execution of CMDM_MACS, the load of the edge network is relatively balanced, that is, the resource utilization of idle ENs is optimized while the execution pressure of individual busy ENs is alleviated.

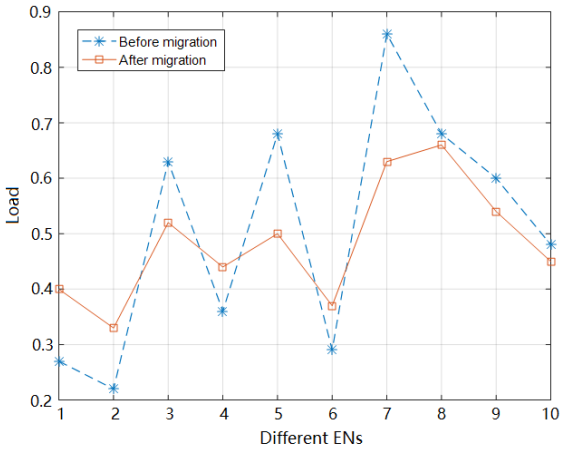

FIGURE 8. Load changing of each EN before and after migration.

In Figure 9, the value of optimization, namely the weighting of load balancing degree and migration cost under the three algorithms is compared. As can be seen that, with the increase of the number of containers, the value of optimization increases slowly. This is because the increase in the number of containers leads to an increase in the resource occupancy rate of ENs, which in turn causes the number of containers to be migrated to increase. The container deployment algorithm based on improved ACS proposed in this paper minimizes the migration cost while taking into consideration both the utilized and remaining resources, and has better performance than GA and SABFD algorithm. Taking 120 containers as an example, the optimization value

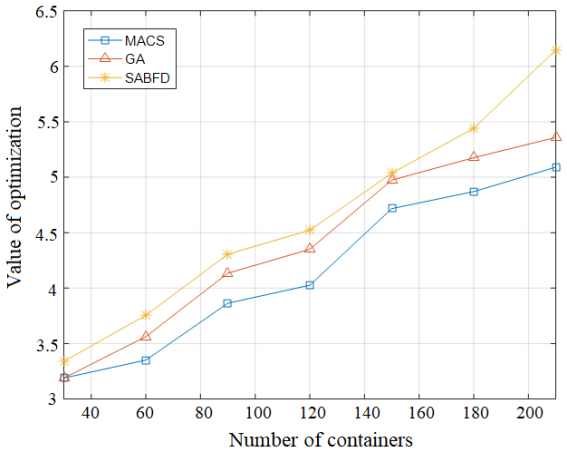

FIGURE 9. Cost of migration with different number of containers under different algorithms.

of MACS algorithm proposed in this paper is reduced by 7.3% and 12.5% respectively compared with GA and SABFD.

VI. CONCLUSION

Aiming at the problem of unbalanced load of edge network caused by different business demands and uneven spatial and temporal distribution of business requests in power IoT, this paper proposes a container migration-based decision-making mechanism for achieving the shunting of business requirements between ENs, thereby minimizing the impact of container migration while realizing the load balancing of edge network and optimizing the resource utilization of EN. The CMDM mechanism divides the container migration problem for load balancing of edge network into three parts. First, determining the timing of container migration based on the load differentiation matrix model between ENs. Then, establishing the container migration model of load balancing joint migration cost to minimize the impact of container migration while balancing the load of edge network. Finally, calculating the migration priority of containers from the perspective of resource correlation and business relevance, and designing the migration algorithm based on improved discrete ant colony system to utilize the migration model and thus guiding the choice of possible migration actions. Simulation results show that the proposed CMDM mechanism can reduce the cost of container migration while improving the load of edge network.

On the basis of three-layer edge network structure, how to realize the optimal dynamic task scheduling for energy consumption, delay and other aspects among ENs by using virtualization technology is an urgent problem to be solved in the future. We will take into consideration the mobility and improve the migration strategy for multiple optimization goals, namely the consumption, load balancing, delay and countless others from the directions of cloud-to-side, side-by-side, and side-to-end collaboration. In addition, performance of MACS algorithm in the actual application based on edge computing in electric power IoT need to be further modified. We will use specific mathematical algorithm to approximate MACS and improve the stability of the algorithm.

REFERENCES

[1] P. Schulz, M. Matthe, H. Klessig, M. Simsek, G. Fettweis, J. Ansari, S. A. Ashraf, B. Almeroth, J. Voigt, I. Riedel, A. Puschmann, A. Mitschele-Thiel, M. Müller, T. Elste, and M. Windisch, "Latency critical IoT applications in 5G: Perspective on the design of radio interface and network architecture," *IEEE Commun. Mag.*, vol. 55, no. 2, pp. 70–78, Feb. 2017, doi: 10.1109/MCOM.2017.1600435CM.

[2] J. Pan and J. McElhannon, "Future edge cloud and edge computing for Internet of Things applications," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 439–449, Feb. 2018, doi: 10.1109/JIOT.2017.2767608.

[3] Z. Tang, X. Zhou, F. Zhang, W. Jia, and W. Zhao, "Migration modeling and learning algorithms for containers in fog computing," *IEEE Trans. Services Comput.*, vol. 12, no. 5, pp. 712–725, Sep. 2019, doi: 10.1109/TSC.2018.2827070.

[4] L. Ma, S. Yi, N. Carter, and Q. Li, "Efficient live migration of edge services leveraging container layered storage," *IEEE Trans. Mobile Comput.*, vol. 18, no. 9, pp. 2020–2033, Sep. 2019, doi: 10.1109/TMC.2018.2871842.

[5] L. Li, J. Dong, D. Zuo, and J. Wu, "SLA-aware and energy-efficient VM consolidation in cloud data centers using robust linear regression prediction model," *IEEE Access*, vol. 7, pp. 9490–9500, 2019, doi: 10.1109/ACCESS.2019.2891567.

[6] F. Farahnakian, T. Pahikkala, P. Liljeberg, J. Plosila, and H. Tenhunen, "Utilization prediction aware VM consolidation approach for green cloud computing," in *Proc. IEEE 8th Int. Conf. Cloud Comput.*, Jun. 2015, pp. 381–388, doi: 10.1109/CLOUD.2015.58.

[7] X.-F. Liu, Z.-H. Zhan, J. D. Deng, Y. Li, T. Gu, and J. Zhang, "An energy efficient ant colony system for virtual machine placement in cloud computing," *IEEE Trans. Evol. Comput.*, vol. 22, no. 1, pp. 113–128, Feb. 2018, doi: 10.1109/TEVC.2016.2623803.

[8] F. Liu, Z. Ma, B. Wang, and W. Lin, "A virtual machine consolidation algorithm based on ant colony system and extreme learning machine for cloud data center," *IEEE Access*, vol. 8, pp. 53–67, 2020, doi: 10.1109/ACCESS.2019.2961786.

[9] M. Lin, J. Xi, W. Bai, and J. Wu, "Ant colony algorithm for multi-objective optimization of container-based microservice scheduling in cloud," *IEEE Access*, vol. 7, pp. 83088–83100, 2019, doi: 10.1109/ACCESS.2019.2924414.

[10] Y. Gao, H. Guan, Z. Qi, Y. Hou, and L. Liu, "A multi-objective ant colony system algorithm for virtual machine placement in cloud computing," *J. Comput. Syst. Sci.*, vol. 79, no. 8, pp. 1230–1242, Dec. 2013, doi: 10.1016/j.jcss.2013.02.004.

[11] D. Puthal, M. S. Obaidat, P. Nanda, M. Prasad, S. P. Mohanty, and A. Y. Zomaya, "Secure and sustainable load balancing of edge data centers in fog computing," *IEEE Commun. Mag.*, vol. 56, no. 5, pp. 60–65, May 2018, doi: 10.1109/MCOM.2018.1700795.

[12] A. Kurve, C. Griffin, D. Miller, and G. Kesidis, "Game theoretic iterative partitioning for dynamic load balancing in distributed network simulation," Tech. Rep., 2012.

[13] M. Jia, W. Liang, Z. Xu, and M. Huang, "Cloudlet load balancing in wireless metropolitan area networks," in *Proc. IEEE INFOCOM 35th Annu. IEEE Int. Conf. Comput. Commun.*, Apr. 2016, pp. 1–9, doi: 10.1109/INFOCOM.2016.7524411.

[14] X. Ma, S. Zhang, W. Li, P. Zhang, C. Lin, and X. Shen, "Cost-efficient workload scheduling in cloud assisted mobile edge computing," in *Proc. IEEE/ACM 25th Int. Symp. Qual. Service (IWQoS)*, Jun. 2017, pp. 1–10, doi: 10.1109/IWQoS.2017.7969148.

[15] L. Lv, Y. Zhang, Y. Li, K. Xu, D. Wang, W. Wang, M. Li, X. Cao, and Q. Liang, "Communication-aware container placement and reassignment in large-scale Internet data centers," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 3, pp. 540–555, Mar. 2019, doi: 10.1109/JSAC.2019.2895473.

[16] T. G. Rodrigues, K. Suto, H. Nishiyama, and N. Kato, "Hybrid method for minimizing service delay in edge cloud computing through VM migration and transmission power control," *IEEE Trans. Comput.*, vol. 66, no. 5, pp. 810–819, May 2017, doi: 10.1109/TC.2016.2620469.

[17] C. Kaewkasi and K. Chuenmuneewong, "Improvement of container scheduling for docker using ant colony optimization," in *Proc. 9th Int. Conf. Knowl. Smart Technol. (KST)*, Feb. 2017, pp. 254–259, doi: 10.1109/KST.2017.7886112.

[18] O. Sukwong, A. Sangpetch, and H. S. Kim, "SageShift: Managing SLAs for highly consolidated cloud," in *Proc. IEEE INFOCOM*, Mar. 2012, pp. 208–216, doi: 10.1109/INFCOM.2012.6195591.
[19] Z. Li and G. Wu, "Optimizing VM live migration strategy based on migration time cost modeling," in *Proc. Symp. Archit. Netw. Commun. Syst. ANCS*, 2016, pp. 99–109, doi: 10.1145/2881025.2881035.
[20] 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, doi: 10.1371/journal.pone.0211729.
[21] S. Wang, R. Urgaonkar, M. Zafer, T. He, K. Chan, and K. K. Leung, "Dynamic service migration in mobile edge-clouds," in *Proc. IFIP Netw. Conf. (IFIP Netw.)*, May 2015, pp. 1–9, doi: 10.1109/IFIPNetworking.2015.7145316.
[22] I. Ahmad, K. Mehrotra, C. K. Mohan, S. Ranka, and A. Ghafoor, "Performance modeling of load-balancing algorithms using neural networks," *Concurrency: Pract. Exper.*, vol. 6, no. 5, pp. 393–409, Aug. 1994.
[23] H. Liu, H. Jin, X. Liao, C. Yu, and C.-Z. Xu, "Live virtual machine migration via asynchronous replication and state synchronization," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 12, pp. 1986–1999, Dec. 2011, doi: 10.1109/TPDS.2011.86.
[24] M. Dorigo and L. M. Gambardella, "Ant colony system: A cooperative learning approach to the traveling salesman problem," *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 53–66, Apr. 1997, doi: 10.1109/4235.585892.
[25] A. Beloglazov and R. Buyya, "Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers," *Concurrency Comput., Pract. Exper.*, vol. 24, no. 13, pp. 1397–1420, Sep. 2012.
[26] S. S. Masoumzadeh and H. Hlavacs, "Dynamic virtual machine consolidation: A multi agent learning approach," in *Proc. IEEE Int. Conf. Autonomic Comput.*, Jul. 2015, pp. 161–162, doi: 10.1109/ICAC.2015.17.
[27] A. Beloglazov, J. Abawajy, and R. Buyya, *Energy-Aware Resource Allocation Heuristics for Efficient Management of Data Centers for Cloud Computing*. Amsterdam, The Netherlands: Elsevier, 2012.
[28] Y. Wen, Z. Li, S. Jin, C. Lin, and Z. Liu, "Energy-efficient virtual resource dynamic integration method in cloud computing," *IEEE Access*, vol. 5, pp. 12214–12223, 2017, doi: 10.1109/ACCESS.2017.2721548.
[29] H. Wang and H. Tianfield, "Energy-aware dynamic virtual machine consolidation for cloud datacenters," *IEEE Access*, vol. 6, pp. 15259–15273, 2018, doi: 10.1109/ACCESS.2018.2813541.

ZITONG MA was born in Xuzhou, Jiangsu, China, in 1995. She is currently pursuing the M.E. degree with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications. Her current research interests include the container migration and edge computing.

SUJIE SHAO (Member, IEEE) received the Ph.D. degree from the Beijing University of Posts and Telecommunication, Beijing, China, in 2015, where he is currently a Lecturer. His research interests include edge computing, the Internet of Things, smart grids, and communication network management.

SHAOYONG GUO received the Ph.D. degree from the Beijing University of Posts and Telecommunication, Beijing, China, in 2013, where he is currently an Associate Professor. His research interests include blockchain, the Internet of Things, ubiquitous networks, and smart grids.

ZHILI WANG is currently an Associate Professor with the Beijing University of Posts and Telecommunications, engaged in research and standardization work in communication networks and computer science and technology. His main research interests include network management, communications software, and interface testing. He won one National Science and Technology Progress Awards and wrote more than eight ITU-T international standards, where he is currently serving as the Working Party 2 Chairman of the ITU-T Study Group 2.

FENG QI received the M.E. degree in computer application from Northeastern University, Shenyang, China, in 1996. He is currently a Professor with the Institute of Network Technology, Beijing University of Posts and Telecommunications, Beijing, China, in 1996, where he is involved in scientific research, teaching, and standardization research in information and communication. He has also written more than ten International Telecommunication Union Telecommunication Standardization Sector (ITU-T) international standards and industry standards. His research interests include communications software, network management, and business intelligence. He was a recipient of two National Science and Technology Progress Awards. He served as the Vice Chairman for the ITU-T Study Group 4 and the Study Group 12.

AO XIONG (Member, IEEE) received the Ph.D. degree in computer science from the Beijing University of Posts and Telecommunications (BUPT), in 2013. He is currently an Associate Professor with the State Key Laboratory of Networking and Switching Technology, BUPT. He has authored about 30 SCI/EI index articles and received 14 national and provincial scientific and technical awards. His research interests include communication networks and communication software, Internet routing and applications, and network management.