Journal of Computers Vol. 34 No. 6, December 2023, pp. 47-62
doi: 10.53106/199115992023123406004

Container Migration Strategy Based on Multi-Objective Optimization for Edge-Cloud Coordination Enabled Smart Grids

Chuqiao Lin[1], Haoran Sun[2], Shengda Wang[2],
Chunyan An[3,4*], Han Qi[2], Xin Luo[3,4]

[1] State Grid Jilin Electric Power Co., Ltd
Jilin, 010010, China
1297213158@qq.com

[2] State Grid Jilin Electric Power Co., Ltd. Information Communication Company.,
Jilin, 010010, China
413309468@qq.com, dollie@126.com

[3] State Grid Smart Grid Research Institute Co., Ltd., Beijing, 102209, China
[4] Electric Power Intelligent Sensing Technology and Application of State Grid Corporation Joint Laboratory,
Beijing, 102209, China
anchunyan@geiri.sgcc.com.cn, xinluo@126.com

Received 30 November 2022; Revised 17 July 2023; Accepted 18 August 2023

Abstract. With the rapid development of information and communication technologies, new services and new applications continue to emerge, especially in smart grid network scenarios. Placing services in terms of containers on the network edge is a promising solution for guaranteeing low latency, large connections, and high bandwidth. In this paper, we propose a multi-objective container migration strategy (MOCMS). Firstly, the container needed to be migrated is selected according to the resource utilization and the energy consumption condition at the network edge. Secondly, in order to avoid the problem of resource fragmentation, the node coordination matrix model is established. Thirdly, in order to obtain the optimal container migration results, an improved Binary Grey Wolf Optimizer (BGWO) algorithm is designed. Finally, the simulation results show that the proposed container migration strategy can perform better than other existing schemes.

Keywords: smart grids, network edge, container migration, multi-objective optimization

1 Introduction

In mobile edge computing (MEC) networks, edge nodes have limited resources. Due to the uneven distribution of service requests of end users, there are obvious differences in the degree of heterogeneity of edge nodes. Fortunately, based on the advantages of rapid container deployment, containerization technology in the microservice architecture can solve the difficulties mentioned in the mobile edge computing network. However, in order to ensure the quality of service for users, the movement of users will cause task migration. The cross-platform capability of containers has become an important support for container migration. Therefore, a reasonable and effective container migration strategy is an important technology for future mobile edge computing networks. By investigating container migration, high resource utilization and resource isolation of mobile edge computing networks can be achieved. Therefore, container migration technology plays an important role in mobile edge computing.

Container migration has caused a wave of research in both academia and industry [1-6]. Literature [1] proposed an optimal migration strategy selection algorithm based on the existing zero downtime migration technology to minimize the response time and network traffic jointly. Literature [2] proposed a migration method based on environment awareness and dynamic compression to reduce downtime in the migration process. Literature [3] proposed a container integration algorithm to reduce the fragmentation of container layout, so as to improve the utilization rate of machine resources effectively. Literature [4] established a two-level scheduling mechanism to monitor global real-time resources and decided whether to proceed with container migration based on the

* Corresponding Author

predicted trend of resource utilization. In order to improve load balancing and reduce migration costs, a decision mechanism and migration algorithm based on container migration were proposed in the literature [5]. Traditional resource scheduling methods lead to high load and energy consumption. Therefore, it is necessary and meaningful to study how to reduce the energy consumption of edge nodes in mobile edge computing networks by reducing the number of running servers through container integration. The existing container migration strategies ignore the overall network condition of the current edge computing network and only rely on the static threshold or linear regression to predict the container migration time.

Therefore, the key research problem of this paper is to solve the problem of edge node resource imbalance and high energy consumption in the mobile edge computing network based on the overall network status of smart grids. In order to solve the above problems, we propose a multi-objective optimization container migration strategy and adopt the Gray Wolf optimizer algorithm to conduct simulation experiments. Firstly, considering the resource utilization of edge nodes, the set of edge nodes with large load differences is obtained through resource balance detection. Secondly, a container migration model was established to jointly optimize resource balance, energy consumption, and migration cost. Then, the target migration node of the container to be migrated is determined by the multi-objective optimization migration strategy. Finally, an improved Binary Grey Wolf Optimizer (BGWO) algorithm based on multi-objective joint optimization is designed to obtain the global optimal container migration result. Designing a container migration strategy based on multi-objective optimization for edge-cloud coordination in smart grids is the technical achievement of this paper. The simulation results verify the effectiveness of the proposed improved algorithm based on the baseline algorithm.

In summary, we propose a multi-objective optimized container migration strategy in smart grids with the improved Gray Wolf algorithm in this paper. The problem of unbalanced edge node resources and high energy consumption in the mobile edge computing network is solved based on the overall network status. Simulation results verify the effectiveness of the proposed improved algorithm.

The structure of this paper is divided into the following sections. Section II presents related work on container migration. Section III describes the system models, including the resource, node coordination, energy consumption, and container migration models. Migration Policy is analyzed in Section IV. Section V describes the Gray Wolf optimization algorithm. Simulation experiments and analysis are presented in Section VI.

2 Related Work

Container technology originated from the open-source platform of Linux. Containers can execute running instructions locally on the CPU without any special interpretation mechanism. Due to the great significance of container migration for edge computing, this research has received a lot of attention from academia and industry. Table 1 lists the current representative literature. Table 1 summarizes the research content of the literature from three aspects: motivation, method, and contribution.

Table 1. Related work

References	Motivation	Method	Contribution
[7]	Explore the continuity of Transmission Control Protocol (TCP) connection microservices.	Overlay network technology to realize TCP connection migration.	The proposed solution does not require the use of a dedicated protocol.
[8]	Explore dependencies between containers and user mobility.	Recurrent neural networks (RNN), Hunger Games Search.	Container migration strategies can effectively reduce service latency and improve server load balancing.
[9]	Solve the compatibility issues of cloud edge systems in hardware and software stacks.	Deep learning.	The proposed method can achieve a 7 times speedup.
[10]	Optimizing virtual network function management reconstruction in the Underlying Network.	Hierarchical auxiliary map.	Optimization management refactoring is effective in the network.
[11]	Dynamic resource allocation in data centers and placement issues to support migration.	Deep learning, unsupervised learning.	Placement and relocation for greater energy efficiency.

[12]	Load balancing, fault tolerance, and system maintenance for resource-constrained edge nodes.	Computational and network-aware resource-sharing framework with dynamic container migration.	Higher system response time gains are obtained.
[13]	Optimal live migration problem for containerized edge computing services.	Migration technology is based on state replication and state reproduction.	The optimal migration method reduces response time and network load.

It can be seen from Table 1 that most of the existing container migration strategies ignore the overall network status of the current edge computing network. Many papers rely only on static thresholds or linear regression to predict container migration times. At the same time, the network needs to consider more comprehensive energy and delay factors [14-17]. Therefore, it is necessary to investigate reducing the energy consumption of edge nodes in mobile edge computing networks by reducing the number of running servers through container integration.

3 System Model

In the MEC-enabled smart grid network scenario, the tasks are offered in terms of containers and deployed at the network edge. As shown in Fig. 1, there are n edge nodes at the network edge, the set of edge nodes is defined as $S = \{S_1, S_2, ..., S_n\}$. Each edge node has the resources of Central-Processing-Unit (CPU), memory, and hard disk, which can be expressed as $R = \{R_1, R_2, ..., R_n\}$. Define a set of m containers as $C = \{C_1, C_2, ..., C_m\}$, each container represents a computation-intensive task. For edge node j, define the resource vector as $R_j = \{r_j^1, r_j^2, ..., r_j^k\}$, where k represents the resource dimension on the edge node, and r_j^k represents the maximum amount of data that can be handled by the *k-th* resource at the node j. The resource utilization rate of edge node j is defined as $U_j = \{u_j^1, u_j^2, ..., u_j^k\}$, where u_j^k represents the resource utilization rate of the resource k on the edge node. The attribution set of container i is defined as $C_i = \{D_i, t_i^{del}, Tol_i\}$, where $D_i = \{d_i^1, d_i^2, ..., d_j^k\}$ represents the resources set required by the container i, d_i^k represents the *k-th* resource required by container i, t_i^{del} represents the computing delay, and Tol_i represents the delay threshold of the task carried by the container. The communication bandwidth between edge node j and j' is B_{jj}.

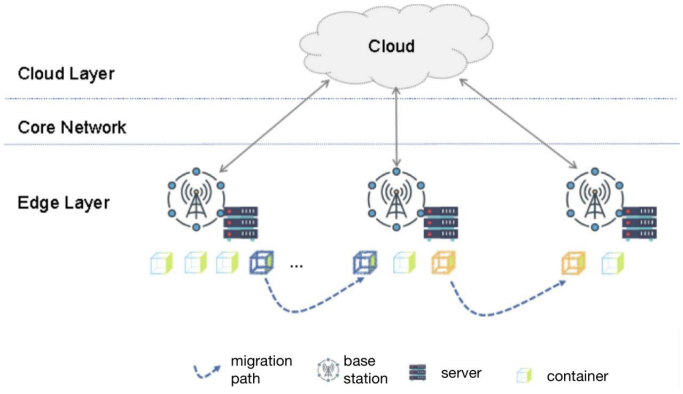

Fig. 1. Architecture of containers migration

3.1 Resource Model

In order to improve the resource utilization of the whole system, the degree of resource fragmentation of edge nodes should be considered. When the computation load at each edge node is too high, the lack of computing resources will degrade the application performance of the container deployed on the edge node. When the load of the edge node is too low, the edge node will be idle, which will increase the energy consumption of the data cen-

ter. The higher the degree of resource fragmentation, the easier it is to generate resource fragmentation.

Define the initial deployment matrix of the container as $X = [x_{ij}]_{I\times J}$, where when the container i is deployed on the edge node j, $x_{i,j} = 1$. Otherwise, the value is 0. x_{ij} is described as:

$$x_{i,j} = \begin{cases} 1, container\ i\ is\ deployed\ on\ edge\ node\ j, \\ 0, container\ i\ is\ not\ on\ edge\ node\ j. \end{cases} \tag{1}$$

Define the set of containers deployed on the edge node S_j as $V(S_j)$. The resource fragmentation of edge nodes refers to the huge difference in the resource utilization rate of different types of resources of nodes. When a certain resource consumes a large amount, the remaining amount of the resource is very small. However, when the consumption of another resource is small, the remaining resource is large. It prevents the edge node from deploying new containers, resulting in the unavailability of remaining resources and waste of resources. Therefore, resource fragmentation occurs when resource edge nodes are easily overloaded. Equation (2) is the utilization rate of the *k-th* resource on the edge node S_j . Equation (3) is the average utilization rate of various resources on the edge node S_j . Equation (4) is the resource utilization of the bit edge node S_j balance degree, where $\Lambda_j \in (0,1)$. The more balanced the edge node resources are, the closer Λ_j is to 0. Otherwise, Λ_j is closer to 1. Equation (5) is the load of the edge node S_j , where γ_k is the load weight of the resource at the edge node. Meanwhile, $\sum_{r_k \in R} \gamma_k = 1$ needs to meet the constraints. The specific constraints are as follows.

$$u_j^k = \frac{\sum_{c_i \in C} x_{i,j} d_i^k}{r_j^k}, \tag{2}$$

$$\overline{u}_j = \frac{\sum_{r_k \in R} u_j^k}{k}, \tag{3}$$

$$\Lambda_j = \sum_{l=1,2\cdots k} \frac{(u_j^l - \overline{u}_j)^2}{k}, \tag{4}$$

$$\Delta_{S_j} = \sum_{r_k \in R} \gamma_k \mu_j^k . \tag{5}$$

This paper determines resource fragmentation from two perspectives, including resource utilization balance and load of a single edge node. The Table 2 shows the algorithm process of resource fragmentation detection:

Table 2. Algorithms for detecting resource fragmentation

Input: (1) $I \times J$ dimensional container deployment matrix X; (2) $K \times 1$ dimensional edge node resource vector R; (3) $K \times 1$ dimensional container resource requirement vector D_i; (4) Constant resource balancing degree c_1, load constant c_2.
Output: Collection of edge nodes with resource fragmentation Φ.
Specific Execution Steps:
1: Initialize: $Y = X$, $\Phi = \varnothing$, $C1 = c_1$, $C2 = c_2$;
2: Calculate the resource balance degree Λ_j of edge nodes S_j according to equation (3-4);
3: Calculated the load Δ_{sj} of edge node S_j is according to Equation (3-5);
4: If $\Delta_{sj} > c_2$ and $\Lambda_j > c_1$, $S_j \in \Phi$.

3.2 Node Coordination Model

In mobile edge computing networks, the load of edge nodes varies with the deployment of different containers. Many types of node resources and a large number of complex tasks easily lead to the difficulty of scheduling among resources. For the same container, different types of resources required by the calculation task of the container are different. For different containers, the demand for certain resources on the edge node is also different. If multiple containers with high demand for certain resources are deployed on the same edge node, some resources on the edge node will consume a lot. However, the remaining resources of other resources are sufficient, resulting in an imbalance of remaining resources of edge nodes. In this case, it is difficult to deploy new containers on the nodes due to the influence of resources with fewer remaining resources. Such phenomena lead to resource fragmentation. In order to reduce resource fragmentation, it is necessary to consider the coordination of different types of containers and edge nodes in the selection of new deployment nodes for migrating containers. In this way, the consumption of various resources on the edge node can be relatively balanced, thereby improving the resource utilization rate of the edge node.

The coordination parameter between container i and edge node j is represented by $corr(i, j)$. The dimensional coordination matrix of N containers and M edge nodes is shown in the formula (6). The range of $corr(i, j)$ is usually between 0 and 1. The closer the value of $corr(i, j)$ is to 1, the resource demand characteristics of container i and edge node j are highly complementary. On the contrary, the smaller the complementary degree of container i and edge node j is.

$$\begin{bmatrix} corr(1,1) & corr(1,2) & \cdots & corr(1,m) \\ corr(2,1) & corr(2,2) & \cdots & corr(2,m) \\ \vdots & \vdots & \ddots & \vdots \\ corr(n,1) & corr(n,2) & \cdots & corr(n,m) \end{bmatrix}_{N\times M}, \tag{6}$$

Similarity measure is a measure to comprehensively evaluate the degree of similarity between two things. The commonly used methods of calculating similarity include calculating distance and angle. In this paper, Euclidean distance is used to calculate the similarity between the resources needed by the container and the resources left by the edge node. In Equation (7), r_j^k represents the quantity of the k -th resource of edge node j, d_i^k represents the required quantity of the k -th resource of container i, and k represents the resource dimension. Equation (8) represents the variance of a dimension resource in the edge node, so the resource coordination based on similarity is Equation (9).

$$d_{i,j} = \sqrt{\sum_{l=1}^{k} \left(\frac{r_j^k - d_i^k}{S_k}\right)^2}, \tag{7}$$

$$S_k = \sum_{l=1}^{j} \frac{(u_j^k - \overline{u}^k)^2}{j}, \tag{8}$$

$$corr(i, j) = \frac{1}{d_{i,j}}. \tag{9}$$

3.3 Energy Consumption Model

Cluster energy optimization can be divided into hardware and software levels. Hardware-level optimization, such as merging traffic, dormant idle switches or links [18], and hot datasets [19], is suitable for cloud service tasks, but not for power simulation of performance computing tasks. Cluster energy optimization based on software method is mainly based on scheduling and migration of virtual machines and containers [20].

There are many factors that affect the system energy consumption. In edge computing network, the energy consumption of data center mainly depends on CPU, memory, disk and other factors. Equation (10) describes the system instantaneous power at time t.

$$P_{dc}(t) = \sum_{i=1}^{n} P_{\mathrm{i}}(t), \tag{10}$$

where $P_i(t)$ is the instantaneous power of the edge node at time t, which can be described as:

$$P_i(t) = \begin{cases} P_i^{idel} + (P_i^{max} - P_i^{idel}) \times U_i(t), & N_e > 0, \\ 0, & N_e = 0, \end{cases} \tag{11}$$

where P_i^{idel} is the idle power consumption of edge node i, P_i^{max} is the maximum power consumption of edge node i, $U_i(t)$ is the CPU utilization of server i at time t, and N_e is the number of edge nodes in the mobile edge computing network.

3.4 Container Migration Model

Container migration can solve the problems of unbalanced resource allocation and high energy consumption in edge computing networks. At the same time, information interaction between edge nodes and containers and container migration cost will be generated during the migration process. In the migration process, the main research objectives of this chapter are to reduce the system energy consumption and balance the resource balance while reducing the migration time as much as possible. In this section, a multi-objective decision model based on migration time, resource balance, node coordination and energy consumption is established to achieve the multi-objective optimization result of reducing migration time and system energy consumption. This chapter divides the migration problem into the following sub-problems:

1) Node overload detection: Select the host to be migrated.
2) Detection of underloaded nodes. After the container migration is finished, the node is placed in sleep state to reduce energy consumption.
3) Select the container to be migrated from the overloaded and underloaded nodes.
4) Multi-objective optimization with different migration strategies.
5) Determine whether to migrate the container and which edge node to migrate the container to.

Define the container migration decision $M_{j,j'}$ as follow:

$$M_{j,j'} = \begin{cases} 1, container\ migrates\ \ from\ edge\ node\ \ j\ to\ \ j', \\ 0, container\ does\ not\ migrate\ \ from\ edge\ node\ \ j\ to\ \ j'. \end{cases} \tag{12}$$

If the value of $M_{jj'}$ is 1, it means that the container is migrated from node j to node j'. Otherwise, $M_{jj'}$ is 0.

The migration time is composed of the data transmission time in the migration process and the downtime of the container. Equation (13) describes the migration time of container i from edge node j to edge node j', where $Data_i$ is the amount of data to be migrated in the migration process of the container, and d_i^{mem} is the container memory. Therefore, the total migration time of all containers is as described in Equation (14).

$$Mig_i^t = \frac{Data_i + d_i^{mem}}{B_{j,j'}}, \tag{13}$$

$$Mig_{cost} = \sum_{c_i \in C} M_{j,j'} Mig_i^t, \tag{14}$$

In this subsection, the resource balancing degree of edge nodes is measured from two aspects: the load balancing degree of edge nodes and the degree of resource fragmentation. The resource balancing degree is shown in Equation (15), where U_b is the load balancing degree and V_b is the degree of resource fragmentation. Equation (16) defines the load balancing degree, where S_i stands for edge nodes and n stands for edge node n. The de-

gree of resource fragmentation is measured by the proportion between the remaining resources of edge nodes. Equation (17) describes the remaining amount of the k -th resource on edge node. Equation (18) describes the degree of resource fragmentation on edge node j. Equation (19) is the residual resource rate of the KTH resource on edge node j. Therefore, the degree of resource fragmentation is described as Equation (20).

$$Balance = U_b + V_b, \tag{15}$$

$$U_b = \sum_{s_i \in S} \frac{(\Delta_{S_i} - \overline{\Delta}_S)^2}{n} \tag{16}$$

$$Re_j^k = R_j^k - \sum_{c_i \in C} x_{i,j} d_i^k, \tag{17}$$

$$V_{s_j} = \sum_{i=1}^{k} \frac{(W_j^i - \overline{W}_j)^2}{k}, \tag{18}$$

$$W = \frac{Re}{R}, \tag{19}$$

$$V_b = \sum_{s_i \in S} V_{s_j}. \tag{20}$$

According to the system energy consumption model, the main factor affecting the energy consumption of the mobile edge computing network is CPU utilization. Equation (10) defines the instantaneous power of the system, so the system energy consumption in time T is expressed as followed.

$$En = \sum_{t}^{T} P_{dc}(t) \times t, \tag{21}$$

To sum up, the multi-objective optimization of container migration can be defined as Equation (22), where α, β, γ satisfies $\alpha + \beta + \gamma = 1$.

$$\Gamma = \alpha Mig_{\cos t} + \beta Balance + \gamma En. \tag{22}$$

In order to reduce the cost of container migration and reduce the energy consumption of the system, the container migration problem can be described as a multi-objective optimization problem P under QoS constraints. Constraint C1 stipulates that the sum of the demands of containers deployed on edge nodes for a certain type of resources should be lower than the maximum capacity of the type of resources. Constraint C2 indicates that the total delay of computing tasks carried by the container to be migrated should not exceed the delay threshold that can be tolerated by the task. Constraint C3 states that a container can only map to an edge node.

$$\begin{aligned} & P: \min\{\alpha Mig_{\cos t} + \beta Balance + \gamma En\} \\ & s.t \quad C1: \sum_{c_i \in C} x_{i,j} d_i^k \le r_j^k, \ \forall s_j \in S, r_k \in R, \\ & \qquad C2: \ t_i^{del} + y_{j,j'}^i Mig_{\cos t} \le Tol_i, \ \forall c_i \in C, s_j \in S, \\ & \qquad C3: \sum_{s_j \in S} x_{i,j} = 1, \ \forall c_i \in C. \end{aligned} \tag{23}$$

4 Migration Policy Analysis

Due to their small size and rapid deployment, container environments are characterized by frequent creation and destruction. Therefore, reasonable container migration can not only make full use of node resources, but also achieve the purpose of energy saving. Using container migration computing to reduce the energy consumption of data centers in mobile edge computing networks, this paper proposes a multi-objective migration strategy based on resource balance, migration time and system energy consumption.

In the container migration process, there are two important steps: the selection of the migration container and the selection of the migration node of the container. MOCMS aims to reduce resource fragmentation while reducing migration costs and system energy consumption. In addition to MOCMS, this chapter also considers other container migration strategies:

1) Random selection (RS) strategy: This strategy is provided by the container cluster management component. The selected migration container can be randomly migrated to the supported edge nodes through the RS strategy.

2) Minimum migration time (MMT) strategy: The downtime of the container is related to the memory size of the container. The migration time of the container is determined by two parts, that is, the amount of data to be transferred during the migration process and the memory size of the container.

3) Resource balancing (RB) strategy: After scheduling, containers are migrated to nodes with load balancing of different types of resources. The goal of the RB strategy is to make full use of all node resources and prevent some nodes from running out of resources while others are idle.

Next, we describe the migration in the container and the selection strategy of the target edge node for container transfer.

Specifically, the container selection strategy can be selected from two aspects: resource correlation and system energy consumption. On the one hand, the policy scheme based on resource dependency has certain advantages. This can significantly reduce the risk of overloading the container's original edge node resources. At the same time, the resource-relevance-biased selection strategy can reduce the migration time of containers. On the other hand, the selection strategy based on system energy consumption will migrate containers to edge nodes with the least energy consumption increase. This selection decision will make each schedule achieve the minimum energy consumption increase and local optimum.

Therefore, the migration probability of container c_i on edge node S_j is based on the migration time. $P_{i,j}^{MMT}$ is denoted as the migration probability of container, which is expressed as Equation (24). λ and μ are the weight factors of the impact of container CPU utilization and memory utilization on migration downtime, which meet $\lambda + \mu = 1$. The expression of $P_{i,j}^{MMT}$ is as follows

$$P_{i,j}^{MMT} = \lambda \frac{d_i^{k_{cpu}}}{\sum_{c_{i'} \in C} x_{i',j} d_j^{k_{cpu}}} + u \frac{d_i^{k_{mem}}}{\sum_{c_{i'} \in C} x_{i',j} d_j^{k_{mem}}}, \tag{24}$$

It is worth noting that minimizing container migration time does not improve the resource balance of edge nodes. Conversely, total migration downtime may increase due to frequent container migrations. Therefore, we employ resource coordination between containers and nodes to measure the impact of containers on edge node resource balance. According to Equation (9), the migration probability of container c_i on edge node S_j based on resource balance is $P_{i,j}^{RB}$, which is denoted as follow

$$P_{i,j}^{RB} = \frac{corr(i,j)}{\sum_{c_{i'} \in C} x_{i',j} corr(i',j)}, \tag{25}$$

It is an admitted fact that container migration involves the migration of service data. The progress of service execution affects the amount of data transferred. The amount of data transferred gradually decreases as the execution progress of the service increases. Therefore, the migration probability of container c_i on edge node S_j based on the amount of migrated data is $P_{i,j}^{TT}$, which is denoted as follow

$$P_{i,j}^{TT} = \frac{t_i}{t_i^{del}}, \tag{26}$$

where t_i is the execution time of container c_i and t_i^{del} is the execution time required by container when computing resources are satisfied. Equation (27) describes the migration priority of container c_i on edge node S_j, where, η, ρ, σ are respectively the weights based on migration time, resource balance degree and migration data amount. Specifically, the parameters η, ρ, σ meet the requirements of $\eta + \rho + \sigma = 1$. According to Equation (27), the priority set of container migration can be obtained as follow

$$P_{i,j} = -\eta P_{i,j}^{MMT} + \rho P_{i,j}^{RB} + \sigma P_{i,j}^{TT}, \tag{27}$$

The selection of target edge nodes of container migration is based on Equation (22). In order to reduce the migration cost of container, the target selection nodes of container to be migrated can be obtained as follow

$$\begin{aligned} &S_j = \arg\min\{\alpha Mig_{\cos t} + \beta Balance + \gamma En\} \\ &s.t\ C1, C2, C3. \end{aligned} \tag{28}$$

In a nutshell, in this subsection, we analyze the Migration Policy. The analysis of the policy mainly includes the migration probability of container based on the migration time, resource balance, and the amount of migrated data. Thus, the priority set of container migration is obtained.

5 Gray Wolf Optimization Algorithm

5.1 Algorithm Principle

In 2014, Mirjalili et al. proposed Grey Wolf Optimizer (GWO) algorithm based on the modeling of Wolf hunting behavior. The core of Gray Wolf algorithm is to use the cooperation between groups to model the Wolf into a mathematical model in the process of predation to solve the optimization problem. This algorithm has strong convergence performance, fewer parameters, simple structure and so on. Moreover, the algorithm has a certain adaptive convergence mechanism which makes it able to avoid local solutions and achieve a balance between local search ability and global search ability. And finally find the exact estimate of the optimal solution obtained in the optimization process.

The gray wolf optimization algorithm simulates the leadership levels and hunting mechanisms of gray wolves in nature. In order to apply the gray wolf optimization algorithm to the mathematical model, it is necessary to construct a gray wolf social class model. The parameters α, β, δ, and ω are denoted to the wolf ranks. The first layer is the leader of the population called α, which is an individual with management ability. It is responsible for decisions about hunting, food distribution and other matters within the group. β assists α in making decisions, and will take over α's position when α of the whole pack becomes vacant. The third layer is δ, which follows the decisions of α and β. The lowest ω is mainly responsible for the balance of relationships within the population. Through the cooperation between populations, the hunting process of Gray wolves can be divided into the following main processes: tracking and approaching prey, chasing and surrounding prey, and attacking prey. The specific mathematical model is as follows:

In the process of hunting, the behavior of rounding up the prey means that the Gray Wolf approaches the prey position. Equation (29) represents the distance between the individual Gray Wolf and the prey. Equation (30) is the formula for updating the position of the Gray Wolf, where t is the number of current iterations, $\vec{A}$ and $\vec{C}$ are coefficient vectors, and $\vec{X}_p$ and $\vec{X}$ are prey position vectors and Gray Wolf position vectors, respectively. The calculation formula of $\vec{A}$ and $\vec{C}$ is as Equation (32) and (33), where $\vec{a}$ is the convergence factor. Equation (31) defines the convergence factor, where $a_{\max} = 2$, $a_{\min} = 0$, and *iteratorNum* is the total number of iterations. As can be seen from Equation (32), with the increase of the number of iterations, $\vec{a}$ linearly decreases from 2 to 0. $\vec{r}_1$ and $\vec{r}_2$ are random numbers modulo [0, 1]. The relevant formulas of the algorithm are expressed as follows

$$|\vec{D}|=|\vec{C}\cdot\vec{X}_P(t)-\vec{X}(t)|, \tag{29}$$

$$\vec{X}(t+1)=\vec{X}_\rho(t)-\vec{A}\cdot\vec{D}, \tag{30}$$

$$\vec{a}=a_{\max}-\left(a_{\max}-a_{\min}\right)\times\left(t\ /\ iteratorNum\right), \tag{31}$$

$$\vec{A}=2\vec{a}\cdot\vec{r}_1-\vec{a}, \tag{32}$$

$$\vec{C}=2\cdot\vec{r}_2, \tag{33}$$

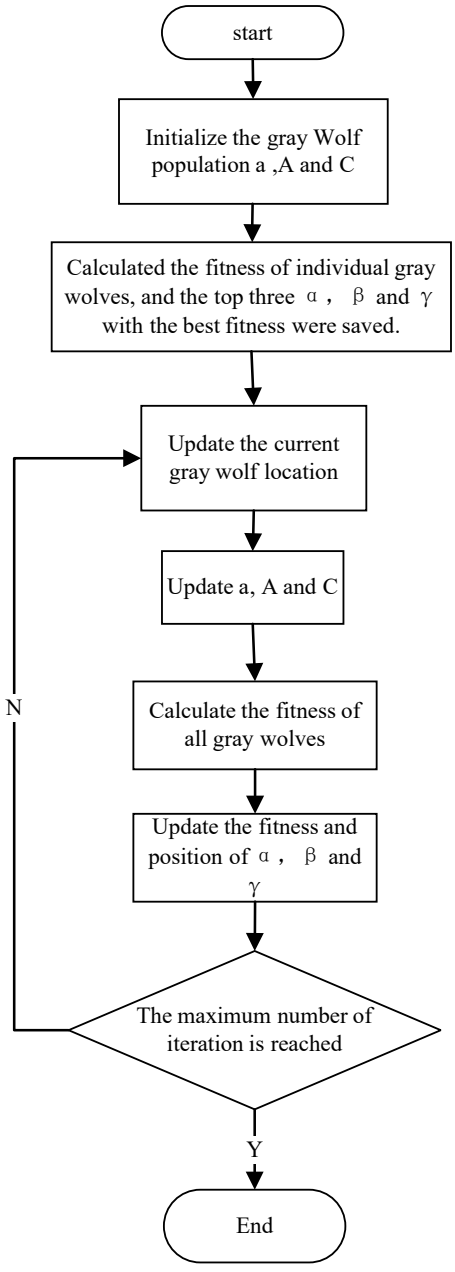

Fig. 2. Flow chart of gray Wolf optimization algorithm

When the prey is located, β and δ wolves are guided by α wolves to surround the prey. The calculation formula of the location of individual Gray Wolf and prey is shown in Equation (34). $\vec{D}_\alpha$, $\vec{D}_\beta$,and $\vec{D}$ represent the distance between α, β, δ Wolf and prey, respectively. $\vec{X}_\alpha$, $\vec{X}_\beta$, and $\vec{X}_\delta$ denote the current position of α, β, δ, respectively. $\vec{C}_1$, $\vec{C}_2$, $\vec{C}_3$ are random vectors and $\vec{X}$ is the current position of the Gray Wolf. In Equations (35), the step length and direction of individual ω in the Wolf pack advancing towards α, β and δ are defined respectively. Equations (36) defines the final position of ω.

$$\begin{cases} \vec{D}_\alpha = \left| \vec{C}_1 \cdot \vec{X}_\alpha - \vec{X} \right|, \\ \vec{D}_\beta = \left| \vec{C}_2 \cdot \vec{X}_\beta - \vec{X} \right|, \\ \vec{D}_\delta = \left| \vec{C}_3 \cdot \vec{X}_\delta - \vec{X} \right|, \end{cases} \tag{34}$$

$$\begin{cases} \vec{X}_1 = \vec{X}_\alpha - A_1 \cdot \vec{D}_\alpha, \\ \vec{X}_2 = \vec{X}_\beta - A_2 \cdot \vec{D}_\beta, \\ \vec{X}_3 = \vec{X}_\delta - A_3 \cdot \vec{D}_\delta, \end{cases} \tag{35}$$

$$\vec{X}(t+1) = \frac{\vec{X}_1 + \vec{X}_2 + \vec{X}_3}{3}. \tag{36}$$

When the prey stops moving, Gray wolves complete the hunting process by attacking behavior. In the process of approaching the prey, the value of $\vec{a}$ gradually decreases linearly from 2 to 0. Meanwhile, the corresponding value of $\vec{A}$ will also change in the interval [-a, a]. When $|\vec{A}|$>1, the location of the next moment away from goal makes the Wolf algorithm has global search ability. When $|\vec{A}|$<1, the wolves to attack their prey, which makes the algorithm showed stronger local search ability. $\vec{C}$ is a vector composed of random values on the interval range [0,2], which means the random weight of the influence of the Wolf pack's position on the prey. $\vec{C}$ >1 means that the influence weight is significant. Otherwise, it means that the influence weight is small. This is beneficial to the random search ability of Gray Wolf algorithm in the search process and avoids the local optimal situation.

Fig. 2 describes the flow chart of the Gray Wolf optimization algorithm. First, the Gray Wolf population and a, A and C are initialized. Then, the fitness of individual Gray wolves in the population was calculated. The top three wolves with the best fitness were kept. At the beginning of the iteration, the positions of the current Gray wolves are updated according to the positions of α, β and δ. The values of a, A and C are updated according to the number of iterations. Then the fitness updates of all Gray Wolf individuals in the population and their positions are calculated until the iteration.

5.2 Algorithm Improvement

For most heuristic algorithms, the diversity of initial solutions is important. The diversity of initial solutions will affect the convergence speed and accuracy of the algorithm. It is worth noting that the gray wolf optimization algorithm is designed to solve the problem of continuous random variables, while the model defined in the container migration process is in binary form. Therefore, the Gray Wolf optimization algorithm needs to be improved to adapt to the optimization of discrete problems. Based on the above considerations, to make the Gray Wolf optimization algorithm more suitable for the container migration model proposed in this paper, the algorithm will be improved from the following two aspects is improved from two aspects:

Initialization: In the process of population initialization, random initialization is generally adopted to generate the initial population. Although this method can quickly obtain the initial solution, it is easy to fall into local optimal solutions when solving complex optimization problems. It is investigated that chaotic variables have the characteristics of uncertainty, regularity and ergodicity. Compared with random sequences, chaotic sequences

can traverse the search space. Besides, to a certain extent, chaotic sequences can avoid falling into local optimal solutions by increasing the search range.

In order to make the initial population diverse, the search space can be traversed as much as possible based on the characteristics of the chaotic sequence. In this paper, the Piecewise mapping function is used to optimize chaos function. The function is shown in Equation (37), where $P \in (0, 1)$ is the control parameter and $x \in (0, 1)$.

$$x_{k+1} = \begin{cases} \dfrac{x_k}{P}, & 0 \le x_k < P, \\ \dfrac{x_k - P}{0.5 - P}, & P \le x_k < \dfrac{1}{2}, \\ \dfrac{1 - P - x_k}{0.5 - P}, & \dfrac{1}{2} \le x_k < 1 - P, \\ \dfrac{1 - x_k}{P}, & 1 - P \le x_k < 1. \end{cases} \tag{37}$$

Binary optimization algorithm of transformation function: according to Equation (12), the container migration decision is a 0-1 decision. Value 1 means that container *i* migrates from edge node *j* to edge node *j*. Value 0 means that the container does not migrate. In the Gray Wolf optimization algorithm, the Wolf pack adopts the continuous position in the global search space to round up the prey, which is used to deal with the continuous problem. However, in binary space, this type of location update does not work. In order to better solve the adaptation problem of Wolf pack algorithm in binary, it is necessary to transform the position information of α, β, and δ between 0 and 1, that is, it is necessary to transform the position of Wolf pack between 0 and 1 with the help of transformation function to solve the discrete problem. The transformation function adopted in this paper is Equation (38), where A is the coefficient vector and D is the distance between individual Gray Wolf and prey.

$$S = \frac{1}{1 + e^{-10*(-A*D-0.5)}} . \tag{38}$$

The improved binary Gray Wolf optimization algorithm process is shown in Table 3.

Table 3. BGWO algorithm flow

Algorithm 2. BGWO algorithm
1: Initialize the gray Wolf population
2: Initialize *a, A, C*
3: Calculate the fitness of each search unit, select the individual with the best fitness X_α, the second best individual X_β, and the third best individual X_δ, and record their positions respectively.
4: While (the end condition is not satisfied)
5: *for i=1 to N do*
6: Calculate the value of attenuation factor a according to Equation (26)
7: Calculate the value of parameters A and C according to Equation (27) and (28)
8: The location of the current search unit X_i is updated according to equations (30) and (31)
9: Calculate the fitness of the current search unit $f(X_i)$
10: end
11: Re-update X_α, X_β and X_δ according to $f(X_i)$
12: end
13: Returns the optimal individual X_α and the fitness function value.

6 Simulation Experiment and Analysis

In this section, Python is used for data simulation experiments. Through the use of a mobile edge network simulation, the performance of BGWO is assessed. By contrasting this algorithm with other algorithms and various migration tactics, its efficacy is demonstrated.

In this experiment, an edge computing scenario with 10 edge servers and 50~200 containers is simulated. The specific parameters are listed in Table 4.

Table 4. Experimental parameters

Parameter name	Parameter value
CPU capacity of the edge node/Core	64
Memory capacity of edge node/GB	[32,64]
Storage capacity of edge nodes/GB	[300,500]
Container CPU resource requirements/Core	[1,4]
Container memory resource requirements/GB	[1,4]
Containers store resource requirements/GB	[2,16]
Node bandwidth/Mbps	[100,300]
The amount of computations for the services hosted by the container/KB	[100,1024]
Service latency threshold/ms	[100,500]

The relationship between the optimization goal value and the number of iterations for various algorithms is compared in Fig. 3. In order to demonstrate the advantages of the Binary Gray Wolf Optimizer (BGWO) proposed in this paper compared to other classic heuristic algorithms, such as Genetic Algorithm (GA) and Ant Colony Algorithm (ACA). The iteration numbers and optimization objective values of the three algorithms are compared. Fig. 3 compares the relationship between the optimization objective value and the number of iterations under different algorithms. On the one hand, it can be seen from the figure that the convergence speed of BGWO algorithm is faster than that of ACA and GA. This is because the BGWO algorithm uses chaotic sequences to initialize the gray wolf population, so that the traversal search of the search space can be realized. On the other hand, it can be seen from Fig. 3 that the target value of the BGWO algorithm is lower than that of the GA algorithm and the ACA algorithm. The smaller the optimization target value, the better the performance of the algorithm. Fig. 3 also reflects the performance characteristics of BGWO over GA and ACA algorithms, as the number of iterations increases. It can be seen from the Fig. 3 that the curve of the BGWO algorithm shows a downward trend as a whole. When the number of iterations is 10, the algorithm falls into a local optimal state as the number of iterations increases. This is because the convergence factor and vector coefficient in the gray wolf population position update are random, which enhances the exploration ability of the algorithm. Finally, the global optimum is reached.

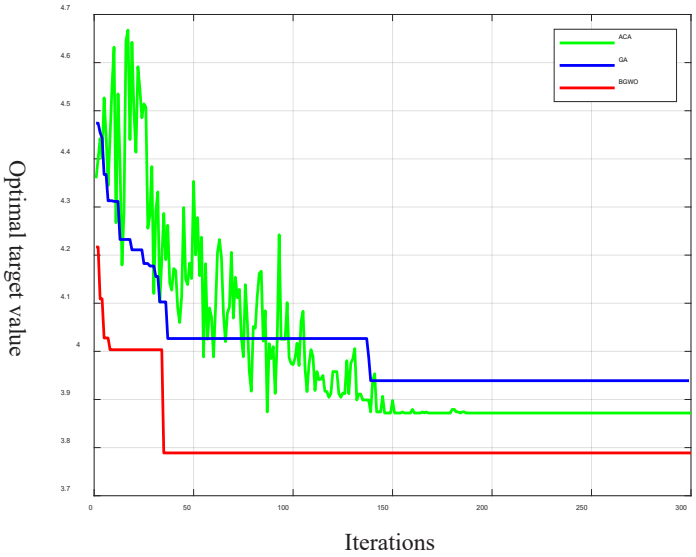

Fig. 3. Relationship between optimization target value and iteration number under different algorithms

Fig. 4 depicts the migration cost of containers under four migration strategies. It is worth noting that the migration cost of containers increases with the number of containers. This is because when the number of containers increases, the utilization of various resources of some edge nodes increases. In this case, the resource balance

of the mobile edge computing network deteriorates. The resource balance of the edge computing network deteriorates, leading to serious resource fragmentation. Therefore, containers need to be migrated to other relatively idle edge nodes for computing, so as to ensure the delay constraints of services carried by containers. It can be seen from the Fig. 4 that the migration strategy proposed in this paper has lower migration cost than strategies based on minimum migration time (MMT), resource balance degree (RB) and random migration (RS). This is because the migration strategy in this paper tends to select edge nodes with low load and high degree of node coordination while considering the minimum migration time. Therefore, the migration strategy adopted in this paper can effectively avoid migration downtime caused by repeated tedious container migrations.

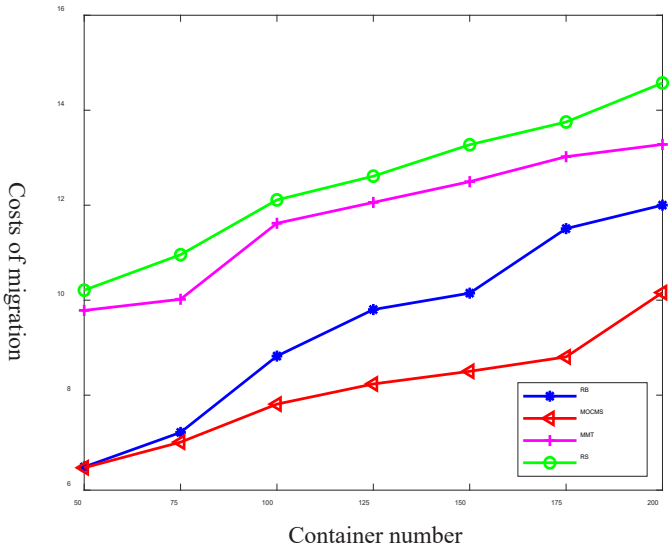

Fig. 4. Container migration costs under different migration policies

Fig. 5 describes the resource balancing under different migration policies. The resource balance degree is measured by the balance degree of the remaining resources of edge nodes and the load balance between edge nodes in the mobile edge computing network. It can be seen from the figure that the strategy proposed in this paper not only reduces the container migration cost and system energy consumption, but also balances the resource balance of nodes.

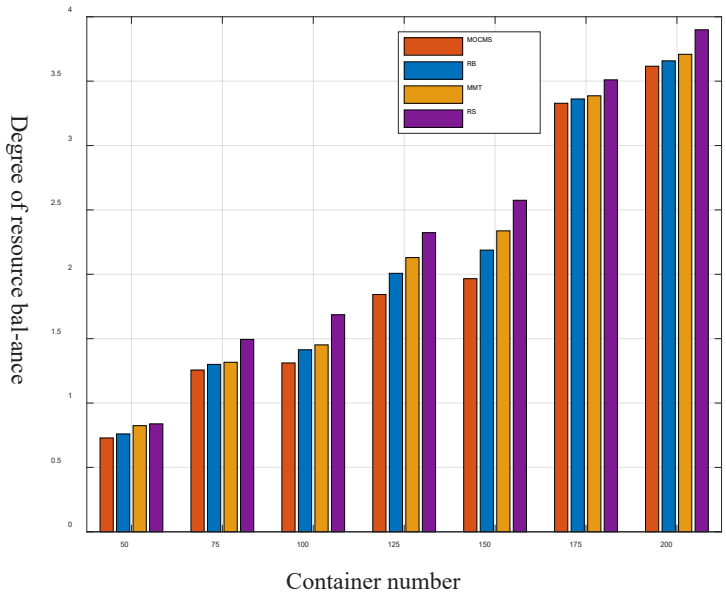

Fig. 5. Resource balancing in different migration policies

7 Conclusion

A container migration strategy based on multi-objective optimization in mobile edge computing networks is investigated in this paper. In order to solve the serious resource fragmentation caused by container migration cost, node coordination and node energy consumption, a multi-objective optimization model is established. Then, the gray wolf optimization algorithm is adopted to reduce resource fragmentation, migration time, and system energy consumption during container migration. However, traditional gray wolf optimization algorithms are suitable for continuous scenarios. In order to make the gray wolf optimization algorithm more suitable for the container migration model in this paper, the transformation function is adopted to transform the wolf position from 0 - 1. At the same time, for the optimization problem, chaotic mapping function is adopted in this paper to optimize the initial population. Simulation results show that the proposed improved gray wolf algorithm significantly improves the convergence accuracy of the algorithm and avoids global errors. Besides, the proposed migration strategy has better performance than other migration strategies. In the future, we will study the hardware implementation of container migration and the consequences of insecure communication.

8 Acknowledgement

This work is supported by State Grid Jilin Electric Power Co., Ltd: Research and Application of Broadband Service Access Trusted WIFI Technology for Transformer Substation (2022-33).

References

[1] T. Kim, M. Al-Tarazi, J.-W. Lin, W. Choi, Optimal Container Migration for Mobile Edge Computing: Algorithm, System Design and Implementation, IEEE Access 9(2021) 158074-158090.
[2] T. Wang, B. Li, W. Chen, Y. Zhang, Y. Han, J. Niu, K. Wong, An Environment-Aware Market Strategy for Data Alloca-tion and Dynamic Migration in Cloud Database, in: Proc. 2019 IEEE 35th International Conference on Data Engineering, 2019.
[3] Z. Zhang, Z. Zhao, H. Li, Static layout and dynamic migration method of a large-scale container, in: Proc. 2021 IEEE 5th Advanced Information Technology, Electronic and Automation Control Conference, 2021.
[4] S. Zheng, F. Huang, C. Li, H. Wang, A Cloud Resource Prediction and Migration Method for Container Scheduling, in: Proc. 2021 IEEE Conference on Telecommunications, Optics and Computer Science, 2021.
[5] K. Li, C. Chang, K. Yun, J. Zhang, Research on Container Migration Mechanism of Power Edge Computing on Load Balancing, in: Proc. 2021 IEEE 6th International Conference on Cloud Computing and Big Data Analytics, 2021.
[6] Y. Yu, A. Calagna, P. Giaccone, C.F. Chiasserini, TCP Connection Management for Stateful Container Migration at the Network Edge, in: Proc. 21st Mediterranean Communication and Computer Networking Conference, 2023.
[7] P. Chhikara, R. Tekchandani, N. Kumar, M.S. Obaidat, An Efficient Container Management Scheme for Resource-Constrained Intelligent IoT Devices, IEEE Internet of Things Journal 8(16)(2021) 12597-12609.
[8] W. Zhang, J. Luo, L. Chen, J. Liu, A Trajectory Prediction-based and Dependency-aware Container Migration for Mobile Edge Computing, IEEE Transactions on Services Computing 16(5)(2023) 3168-3181.
[9] J. Kennedy, V. Sharma, B. Varghese, C. Reaño, Multi-Tier GPU Virtualization for Deep Learning in Cloud-Edge Systems, IEEE Transactions on Parallel and Distributed Systems 34(7)(2023) 2107-2123.
[10] T. Li, Z. Zhu, QoS-Aware Management Reconfiguration of vNF Service Trees With Heterogeneous NFV Platforms, IEEE Transactions on Network and Service Management 20(2)(2023) 2013-2024.
[11] S.A. Khan, M. Abdullah, W. Iqbal, M.A. Butt, F. Bukhari, S.-U. Hassan, Automatic Migration-Enabled Dynamic Resource Management for Containerized Workload, IEEE Systems Journal 17(2)(2023) 2378-2389.
[12] F. Barbarulo, C. Puliafito, A. Virdis, E. Mingozzi, Enabling Application Relocation in ETSI MEC: A Container-Migration Approach, in: Proc. 2022 IEEE 33rd Annual International Symposium on Personal, Indoor and Mobile Radio Communications, 2022.
[13] S. Choudhury, S. Maheshwari, I. Seskar, D. Raychaudhuri, ShareOn: Shared Resource Dynamic Container Migration Framework for Real-Time Support in Mobile Edge Clouds, IEEE Access 10(2022) 66045-66060.
[14] P. Shantharama, A.S. Thyagaturu, A. Yatavelli, P. Lalwaney, M. Reisslein, G. Tkachuk, E.J. Pullin, Hardware Acceleration for Container Migration on Resource-Constrained Platforms, IEEE Access 8(2020) 175070-175085.
[15] S.A. Khan, M. Abdullah, W. Iqbal, M.A. Butt, F. Bukhari, S.-U. Hassan, Automatic Migration-Enabled Dynamic Resource Management for Containerized Workload, IEEE Systems Journal 17(2)(2023) 2378-2389.
[16] Z. Ma, S. Shao, S. Guo, Z. Wang, F. Qi, A. Xiong, Container Migration Mechanism for Load Balancing in Edge Network Under Power Internet of Things, IEEE Access 8(2020) 118405-118416.

[17] C. Rong, J. H. Wang, J. Liu, T. Yu, J. Wang, Exploring the Layered Structure of Containers for Design of Video Analytics Application Migration, in: Proc. 2022 IEEE Wireless Communications and Networking Conference, 2022.
[18] P. Mahadevan, P. Sharma, S. Banerjee, P. Ranganathan, Energy aware network operations, in: Proc. 2009 IEEE International Conference on Computer Communications, 2009.
[19] E. Pinheiro, R. Bianchini, Energy Conservation Techniques for Disk Array-Based Servers, in: Proc. ICS '04: Proceedings of the 18th annual international conference on Supercomputing, 2004.
[20] L. Baresi, S. Guinea, G. Quattrocchi, D.A. Tamburri, MicroCloud: A Container-Based Solution for Efficient Resource Management in the Cloud, in: Proc. IEEE International Conference on Smart Cloud, 2016.