Availability-Constrained Application Deployment in Hybrid Cloud-Edge Collaborative Environment

Wei Xu[1,2], Bing Tang[1,2(✉)], Feiyan Guo[1,2], and Xiaoyuan Zhang[1,2]

[1] School of Computer Science and Engineering, Hunan University of Science and Technology, Xiangtan 411201, China
[2] Hunan Key Laboratory for Service Computing and Novel Software Technology, Hunan University of Science and Technology, Xiangtan 411201, China
btang@hnust.edu.cn

Abstract. Cloud computing offers strong availability and lower cost, while edge computing has lower delay. Deployment of applications by placing microservices in containers in a cloud-edge collaborative environment is adopted by more and more enterprise application providers. For users, they care more about application response time and application availability. For application providers, they also need to save deployment costs to the maximum extent. Therefore, the application deployment in hybrid cloud-edge collaborative environment is a multi-objective optimization problem. In this paper, a genetic algorithm named DP-GA based on improved NSGA-II has been proposed to solve the multi-objective NP-hard problem. We balance the two objectives of minimizing deployment cost and average response time under availability constraints. Using the real dataset of Shanghai Telecom, the experimental results show that the proposed DP-GA is superior to the existing methods, reducing average response time by about 35% and saving deployment cost by about 15%.

Keywords: Composite application · Microservice deployment · Availability · Genetic algorithm · Cloud-edge collaboration

1 Introduction

With the advent of the era of the Internet of Everything and the advent of the 5G era, the data generated by network edge devices are growing rapidly. At this time, the centralized processing in the cloud computing center will not be able to efficiently process the data generated by edge devices. With the wide application of edge computing, some problems are constantly exposed, such as more expensive deployment costs and poor availability. Therefore, this paper discusses deploying applications in the cloud-edge collaboration environment, taking advantage of the two computing modes, cloud computing and edge computing.

The application deployed in this paper is based on the microservice architecture. In recent years, the microservice architecture has received extensive

H. Gao et al. (Eds.): CollaborateCom 2022, LNICST 460, pp. 233–248, 2022.
https://doi.org/10.1007/978-3-031-24383-7_13

attentions. Through combining microservices to build composite applications, it brings strong scalability and flexibility, which has become the deployment method widely chosen by many application providers. In this paper, we encapsulate microservices through containers, and then deploy the containers in virtual machines (VM) which are hosted by physical machines (PM). Both edge data center and cloud data center can deploy microservices.

Based on traditional cloud computing, due to its large response delay, it is urgent to shorten the application response time, especially for applications that require timely response. At the same time, edge computing has relatively poor availability and high deployment costs. Therefore, we combine the needs of both application providers and users, and we mainly focus on the following issues:

(1) The availability of edge data centers closer to the city center is worse due to electromagnetic interference in the city center and higher frequency of user access. At the same time, compared with edge computing, cloud computing often has higher availability due to centralized management.
(2) Because edge data center is often located in the urban area, its management cost is relatively high, and cloud data center is often far away from the urban area, centralized management cost is usually relatively low.
(3) If the microservice is deployed in an edge data center that is closer to the city center, the closer it is to the user, the shorter the response time will be, and the farther it is from the user center, the higher the response time. In addition, the response time of the cloud computing center is often the farthest from the user and has the highest application response time [3,11].
(4) When deploying microservices, we should consider whether the deployed microservices encapsulated in containers exceed the maximum resource capacity of virtual machines.

In response to the above problems, under the conditions of availability and resource constraints, we study minimizing deployment cost and minimizing application response time based on the microservice architecture. It is an NP-hard problem. We propose a new algorithm named DP-GA based on the genetic algorithm to realize the optimal application deployment in the hybrid cloud-edge collaborative environment.

The main contributions of this paper are summarized as follows:

– First, we define the optimal application deployment problem in the cloud-edge collaborative environment as a constrained multi-objective optimization problem. The goal is to deploy composite applications under resource constraints and availability constraints, so as to minimize the response time and deployment cost.
– Secondly, we propose a new method based on genetic algorithm to solve the multi-objective application deployment problem for composite applications, which is characterized by a specially designed population initialization method to obtain a better initial population to achieve better experimental results.
– Finally, to evaluate the proposed method, we obtain multiple Shanghai edge data centers and Western China Cloud Computing Center through clustering

algorithm using the real Shanghai Telecom dataset to construct a cloud-edge collaborative environment. We compared our method with several existing algorithms by changing the number of microservices, and the availability constraint coefficient. The experimental results show that our method outperforms the existing methods.

The remainder of this paper is organized as follows. Section 2 discusses the related work on microservice allocation in container-based edge clouds and existing microservice deployment in cloud-edge environments. Section 3 defines the problem model. Section 4 introduces the improved algorithm DP-GA. Section 5 describes the experimental results and analysis. Section 6 summarizes the whole paper and discusses future work.

2 Related Work

This section presents related work on container-based microservice allocation and existing microservice deployment in cloud-edge environments. The main challenges that need to be addressed in our problem are also highlighted.

In recent years, cloud computing is emerging as a hosting model for delivering business applications. How to deploy application services with the best quality of service becomes a key issue [10,13]. Several research efforts have investigated the service deployment problem. For example, Wen et al. studied application deployment on federated clouds to minimize deployment costs while meeting the security and reliability requirements of deployment [13]. Shi et al. minimized the product of deployment cost and response time of business applications through an algorithm based on genetic algorithm [12], and there are also some multi-objective studies such as network transmission time [4,6,8,9], load balancing or energy consumption [1,4–6,8,9], availability [1,6] and other similar requirements [8]. However, most of these studies consider the problem of microservice allocation in cloud environments. With the continuous development of the Internet era and the arrival of the 5G era, Internet mobile devices and smart wear are becoming more and more popular, and in some situation, cloud computing can not satisfy the requirements of rapid application response without the assistance of the edge computing. Focusing on cloud computing alone will not suffice. The research in this paper is based on the deployment of microservices in the cloud-edge collaborative environment.

The research on container-based microservice deployment of composite applications considers a wide range of goals. Most of the works apply multi-objective optimization algorithms such as Ant Colony Algorithm (ACO) [6], NSGA-II [2], Particle Swarm Optimization (PSO) [5] to solve the problem. However, since these works are conventional multi-objective algorithms, in order to achieve better results and deal with more complex multi-layer (microservice-container-VM-PM) deployment problems, we intend to develop a new multi-objective algorithm to solve microservice multi-instance deployment problem of multiply composite applications in container-based cloud-edge collaborative environment.

3 Problem Description

3.1 Resource Model

In this paper, we study the deployment of composite application in the cloud-edge collaborative environment. The problem is that a set of composite applications $CA = \{ca_1, ca_2, ..., ca_i, ..., ca_s\}$, where ca_i represents the i-th composite application, s represents the number of composite applications, and each application consists of a set of microservices $MS = \{ms_1, ms_2, ..., ms_l, ..., ms_o\}$, where ms_l represents the l-th microservice, and o represents the number of microservices. Here, we assume that each microservice has two instances, and each instance is mapped to the container $C = \{c_{11}, c_{12}, c_{21}, c_{22}, ..., c_{l1}, c_{l2}, ..., c_{o1}, c_{o2}\}$, where c_{l1}, c_{l2} means that the two instances of the l-th microservice are mapped to the corresponding two containers, and c_{lj} is used to represent a specific container, where $j \in \{1, 2\}$. The memory capacity of each container is $\alpha(c_{lj})$.

There is a set of virtual machine types $\Psi = \{\psi_1, \psi_2, ..., \psi_q, ..., \psi_t\}$ used to allocate containers, where ψ_q represents the q-th virtual machine type, and t represents the total number of virtual machine types. For the type of virtual machine selected by the cloud-edge data center to host the container, we use $VM = \{vm_1, vm_2, ..., vm_p, ..., vm_x, vm_{x+1}\}$ to represent, where vm_p denotes the type of virtual machine selected when the p-th edge data center allocates the container, x represents the total number of edge data centers, and vm_{x+1} represents when the virtual machine type selected when the cloud data center needs to host containers. The capacity of each virtual machine is represented by $\delta(vm_p)$. The cost corresponding to each virtual machine is represented by $VMF = \{vmf_1, vmf_2, ..., vmf_p, ..., vmf_x, vmf_{x+1}\}$, where vmf_p represents the cost of the p-th virtual machine.

We define $DC = \{dc_1, dc_2, ..., dc_p, ..., dc_x, dc_{x+1}\}$ to represent the cloud-edge data center location set, where dc_p represents the location of the p-th edge data center, x represents the total number of edge data centers, dc_{x+1} represents the specified cloud data center location, and the corresponding capacity of PM in this data center is represented by $\Lambda(dc_p)$. Depending on the location of the data center, the availability is different. The availability of the cloud computing center is determined by the SLA provided by the VM, and the edge data center determines its availability by a given PM. For the convenience of calculation, the failure rate of a set of edge data centers and cloud computing centers can be defined as $F = \{f_1, f_2, ..., f_p, ..., f_x, f_{x+1}\}$, where f_p represents the failure rate of the PM of the p-th edge data center, and f_{x+1} represents the failure rate of VMs in the cloud data center. The set of user center locations is represented by $UC = \{uc_1, uc_2, ..., uc_k, ..., uc_r\}$, where uc_k represents the k-th user center in the city, and r represents the total number of user centers.

3.2 Time Model

The total amount of work of the i-th composite application ca_i is determined as: $tw_i = \sum_{k=1}^{r} r_{ik}$, where r_{ik} represents the request rate of the k-th user center of

the application ca_i. Each service instance is modeled as a $M/M/1$ queue, since a service can be shared by multiple applications, service aggregation workloads can be accessed through

$$wl = \sum_{i=1}^{s} tw_i \qquad s.t. \quad ms_l \in ca_i \tag{1}$$

For a VM instance with a processing capacity of ϕ, according to Little's law [7], the average request time for a service ms_l is

$$st_l = \frac{1}{\phi - wl} \tag{2}$$

Average latency between user center and the i-th application is

$$ua_i = \frac{\sum_{k=1}^{r} r_{ik}(Bt_{ik} + Et_{ik})}{tw_i} \tag{3}$$

where Bt_{ik} represents the delay from the k-th user center to the i-th application's starting service, Et_{ik} represents the network delay from the i-th application's ending service to the k-th user center.

We assume that mst_{ab} represents the network latency between the service mst_a and its successor service mst_b, and ua_i, mst_{ab} are usually determined by several immutable factors in the communication network. So, ART_i is calculated by $ART_i = ua_i + MS(ca_i)$, where $MS(ca_i)$ refers to the response time of the ca_i in the workflow. We define two functions EST (Earliest Start Time) and FT (Finish Time) for each service to calculate the application makespan. The computing process begins with the starting service, and the MS is the finish time of the ending service.

$$\begin{aligned} &EST(start_i) = 0 \\ &FT(ms_l) = EST(ms_l) + st_l \\ &EST(ms_l) = \max_{ms_a \in Pre(ms_l)} \{FT(ms_a) + mst_{al}\} \\ &MS(ca_i) = FT(end_i) \end{aligned} \tag{4}$$

So, the average response time TRT is expressed by:

$$TRT = \frac{\sum_{i=1}^{s} tw_i ART_i}{\sum_{i=1}^{s} tw_i} \tag{5}$$

3.3 Cost Model

We define a binary variable D_{lj}^{p} to represent the location of the cloud-edge data center where the container is deployed.

$$D_{lj}^{p} = \begin{cases} 1 & \text{if } c_{lj} \text{ is placed on } dc_p \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

We define the cost of deploying the microservice ms_l as

$$msf_l = \sum_{j=1}^{2} \sum_{p=1}^{x+1} vmf_p D_{lj}^p \tag{7}$$

So, the total cost of composite application deployment is calculated through

$$Cost = \sum_{l=1}^{o} msf_l \tag{8}$$

3.4 Availability Constraints

The microservice ms_l obtains the failure rate of the corresponding container where it is deployed:

$$Dep(f_p) = \begin{cases} f_p & \text{if } (\sum_{j=1}^{2} \sum_{p}^{x+1} D_{lj}^p) > 0 \\ 1 & \text{else} \end{cases} \tag{9}$$

Then, the availability of the corresponding microservice ms_l can be defined as

$$Msa_l = 1 - \prod_{p=1}^{x+1} Dep(f_p) \tag{10}$$

The availability of application ca_i can be defined as

$$A_i = \prod_{l=1}^{o} Msa_l \quad s.t. \quad ms_l \in ca_i \tag{11}$$

The average availability for a set of composite applications is

$$Availability = \frac{\sum_{i=1}^{s} A_i}{s} \geq a \tag{12}$$

where a is the minimum availability specified by the application provider, and we use feasible solutions under availability constraints to satisfy the availability constraints as reference solutions:

$$a = A_{minimum} + k(A_{maximum} - A_{minimum}) \tag{13}$$

Among them, $A_{maximum}$ and $A_{minimum}$ are the highest availability and the lowest availability among various schemes deployed. k is the strictness coefficient that controls availability, and $k \in [0, 1]$. The larger the k, the more stringent the availability specified by the application provider.

3.5 Resource Constraints

It is required that the total resources of all containers allocated to each created virtual machine cannot exceed its capacity, and for each virtual machine type vm_p selected in the cloud-edge data center, it must satisfies

$$\sum_{l=1}^{o} \sum_{j=1}^{2} \alpha(c_{lj}) D_{lj}^p \leq vm_p \tag{14}$$

At the same time, it also needs to satisfy that the total capacity of the virtual machines created by each PM does not exceed the corresponding PM's capacity. For the PM's capacity $\Lambda(dc_p)$ of each cloud-edge data center, it must satisfies

$$vm_p \leq \Lambda(dc_p) \tag{15}$$

3.6 Problem Definition

Here, a dual-objective optimization problem is defined to solve the optimal deployment under the conditions of minimizing cost and delay, subject to availability constraints and resource constraints, as shown in Eq. (16).

$$\begin{aligned} &\min Cost = \textstyle\sum_{l=1}^{o} msf_l \\ &\min TRT = \frac{\sum_{i=1}^{s} tw_i ART_i}{\sum_{i=1}^{s} tw_i} \\ &s.t. \quad Availability = \frac{\sum_{i=1}^{s} A_i}{s} \geq a \\ &\quad \textstyle\sum_{l=1}^{o} \sum_{j=1}^{2} \alpha(c_{lj}) D_{lj}^{p} \leq vm_p \\ &\quad vm_p \leq \Lambda(dc_p) \end{aligned} \tag{16}$$

4 Algorithm Implementation

This section proposes a microservice deployment method based on cloud-edge collaboration. Firstly, the K-means clustering algorithm is used to cluster and group the edge server coordinates, and the edge servers in the same cluster are clustered to the same location, which can greatly reduce the search space. Then, an improved multi-objective genetic algorithm DP-GA based on NSGA-II is used to solve the microservice deployment to achieve a balance between dual objectives under constraints.

4.1 Chromosome Coding

We use chromosomes to encode service deployment solutions, and use an array of real integers to represent microservice placement strategies. Each chromosome represents a deployment strategy. Each code on each chromosome represents the location where each microservice is deployed, which carries the container where the microservice is located and its corresponding VM.

4.2 Fitness Function and Constraints

The fitness function is used to judge the pros and cons of individuals, each solution is represented by each individual, and all solutions constitute the overall population. The fitness function in this paper includes two categories: TRT and $Cost$. In addition to the fitness function, the constraints in this paper include two aspects. The first is the availability constraint. Any deployment scheme must meet the availability constraints. The closer to the city center, the higher the cost, the shorter the response time, and the worse of the availability. Secondly,

resource constraints must be met. The virtual machine hosted by each physical machine must not exceed the capacity of the physical machine, and the resource of the container hosted by the virtual machine must not exceed the available resource of the virtual machine.

4.3 Population Initialization

The purpose of initialization is to create a set of different solutions. First, we heuristically assign the containers containing microservices to a set of VMs with random types (uniformly selected from the VM table) using First Fit (FF). Then, use FF to assign the VM to the PM. The use of FF guarantees an efficient solution as well as a unified VM/PM allocation problem.

In general, GAs randomly generate an initial population to ensure the diversity of the population. In order to improve the solution quality and convergence speed, this paper firstly uses the Differential Evolution Algorithm (DE) to determine a part of the population, and the rest of the population will be randomly initialized. The differential evolution algorithm is an efficient global optimization algorithm to prevent falling into a local optimum. By determining a part of the population first through the differential optimization algorithm, the quality of the evolutionary solution can be greatly improved.

4.4 Selection Operator

The improved genetic algorithm DP-GA selection operator in this paper adopts the elite reserved tournament selection operator (etour). Through the replacement sampling method, it is guaranteed that the optimal individual will be selected to participate in the tournament. This method can retain the optimal individual to the next generation of population until the matching pool is sufficient.

4.5 Crossover Operator

The crossover operator adopts the simulated binary crossover (recbx), which adopts the single-point crossover method. The selection of the crossover point is random, and the selection range of the crossover point is from 1 to the number of genes on each chromosome. Small damage can better maintain excellent individuals.

4.6 Mutation Operator

The mutation operator is to adopt two point swapping mutation (mutswap). After determining the deployment location of the microservice, we also determine the cost and response time of the microservice deployment. Since the composite application deployment problem in this paper is modeled as having to the acyclic graph (DAG) form, fully consider the dependence between services, the response

time and deployment cost of different services are different when the containers are deployed in different locations. It does not destroy excellent individuals, reduces the probability of rapid convergence, and can continuously seek the optimal two goals through mutation.

4.7 Algorithm Description

Our proposed DP-GA algorithm, as shown in Algorithm 1, follows the standard framework of NSGA-II, a multi-objective genetic algorithm. The algorithm initializes the population through the differential evolution algorithm and the random generation combination method, and the solution is represented as a group of containers containing microservices corresponding to the location number of the VM where the VM is located. The pareto dominance relationship is obtained by fast non-dominated sorting, and then the priority between individuals is determined by calculating the crowding degree, so that the individuals in the quasi-pareto solution can be extended to the entire pareto domain, and evenly distributed to maintain the diversity of the population, the elite strategy is introduced, the sampling space is expanded, the parent population and the child population are merged to ensure that the excellent individuals can be retained, and then iteratively iterates through the crossover and two-point exchange mutation methods. This evolutionary process ends when a predetermined number of generations or a satisfactory fitness value level is reached.

Algorithm 1. Genetic algorithm DP-GA based on NSGA-II.

Input: A set of microservices
Output: The allocation of microservices

Initialize a population P with individuals;
while Termination Condition is not meet **do**
    for each individual **do**
        evaluate the fitness values;
    end for
    while children number is less than the population size **do**
        apply binary elite tournament selection to select two parents;
        apply simulated binary crossover over the selected parents;
        apply two point swapping mutation on two children;
        add the children into a new population U;
    end while
    evaluate individuals from U;
    non-dominated sorting of $P \cup U$;
    calculate crowding distance of $P \cup U$;
    $P \leftarrow$ select population size of individuals from $P \cup U$;
end while
return the Pareto front of solutions;

5 Experimental Results and Analysis

In this paper, we study the deployment of applications in a cloud-edge collaboration environment based on availability and resource constraints, measuring and evaluating average response time and deployment costs. We compare our proposed algorithm with three other algorithms including differential evolution algorithm (DE), ant colony algorithm (ACO), and particle swarm optimization algorithm (PSO).

5.1 Dataset

The experiment is conducted using the real dataset of Shanghai Telecom, which contains the locations of around 3,000 base stations. We cluster the base stations through a K-means clustering algorithm to form 27 edge data centers. Together with the Western China Cloud Computing Center, we obtain a total of 28 cloud-edge data centers which are used as locations for microservice deployment. The center of the 16 districts in Shanghai is taken as the location of user centers. Here, it is assumed that the total number of visits in the whole Shanghai city is 100 times/second, according to the proportion of the population of each district, we calculated the visit frequency of each user center, as shown in Table 1.

Table 1. User center visit frequency.

User centers	User center visit frequency (times/s)
Huangpu District	2.7
Xuhui District	4.5
Changning District	2.8
Jingan District	3.9
Putuo District	4.9
Hongkou District	3
Yangpu District	5
Minhang District	10.7
Baoshan District	9
Jiading District	7.4
Pudong New Area	22.8
Jinshan District	3.3
Songjiang District	7.7
Qingpu District	5.1
Fengxian District	4.6
Chongming District	2.6

According to the location of each data center and the distance from the city center, the cost of each virtual machine type and the failure rate of the physical machine in each data center are determined. The closer to the city center, the higher the virtual machine rental cost. Due to the interference of various signals in the city center and the high frequency of user visits, the corresponding failure rate is also higher. Here, the Western China Cloud Computing Center has the

Table 2. Virtual machine price and failure rate.

Data center location	Virtual machine price ($/month)			Failure rate
	1Core 1G	2Core 2G	2Core 4G	
Huangpu District	12	23	45	0.047
Xuhui District	10.2	20.1	39.6	0.037
Changning District	11.5	22.7	43.9	0.043
Jingan District	10.6	20.8	40.2	0.039
Putuo District	7.9	15.8	30.9	0.02
Hongkou District	9.1	17.8	36.5	0.028
Yangpu District	9.7	19.2	37.7	0.032
Minhang District	8.7	17.2	33.9	0.025
Baoshan District	9.5	18.6	36.9	0.03
Jiading District	9.9	19.4	38.1	0.035
Pudong New Area	10.9	21.3	41.7	0.041
Jinshan District	7	13.8	26.5	0.01
Songjiang District	7.3	14.1	27.5	0.016
Qingpu District	8.3	16.2	32.1	0.022
Fengxian District	7.7	14.9	29.3	0.018
Chongming District	6.8	13.1	25.7	0.009
Western China Cloud Computing Center	5.4	10.2	19.6	0.0000001

lowest virtual machine rental cost, the lowest failure rate, and is the farthest from the city. Table 2 shows the price of different virtual machine types and the failure rate for the edge data centers in Shanghai and the Western China Cloud Computing Center.

5.2 Simulation Settings and Parameter Settings

Table 3. Parameter value for GA.

Key parameter	Value
Crossover probability	0.9
Mutation probability	0.1
The maximum number of iterations	100
Population proportion of differential evolution algorithm	20%
The number of populations	500
Time delay of each microservice processing request	[3 ms, 6 ms]

In our experiment, we simulate three application and 10 microservices as shown in Fig. 1. We assume that each microservice can be loaded and run with the virtual machine type of "1Core1G", which is the minimum environment to run a microservice.

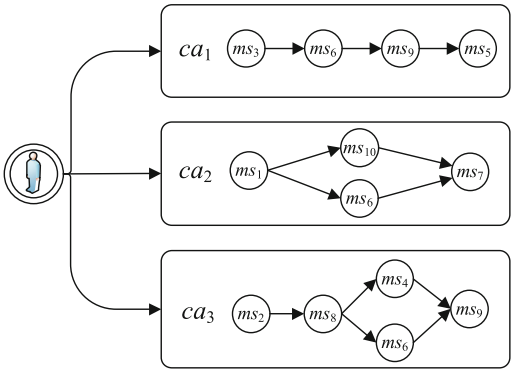

Fig. 1. Three applications with 10 microservices in our experiments.

Assuming that all requests are sent by user centers in Shanghai, and the total number of requests is fixed. According to the population proportion of each user center, the number of requests sent by each user center is calculated. Experiment environment are described as follows: the operating system is Windows 11; CPU is Intel Core i7-8700 with 3.2 GHz; the memory is 16 GB. The programming language is Python, and NSGA-II is implemented using Python Geatpy package. The values of the parameters for genetic algorithm we set are shown in Table 3.

5.3 Results and Analysis

Comparison of TRT and Cost Under Different Numbers of Microservices. We take the application ca_1 as an example, and change the number of deployed microservices. The minimum number of microservices is 3, the maximum number is 7, and the step size is 1. Each microservice is deployed in different cloud-edge data centers, and the availability constraint is fixed at 92%. DP-GA is compared with PSO, ACO and DE, in terms of the average response time and deployment cost.

First of all, the comparison results of average response time are shown in Fig. 2. It can be seen from this figure that no matter how many microservices there are, the TRT obtained by DP-GA is the smallest and it is about 35% better than other algorithms which shows that DP-GA is optimal. At the same time, it shows that when the number of microservices reaches 7, the TRT is greatly increased. The increase is due to the fact that when there are 7 microservices and the constraint is 92%, the cloud computing center participates in the deployment of microservices to increase the availability and reduce the failure rate.

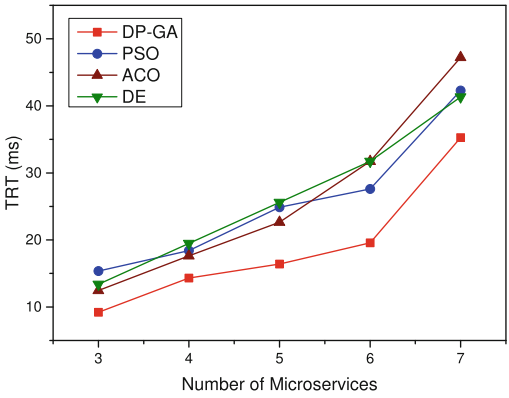

Fig. 2. Comparison of average response time with different numbers of microservices.

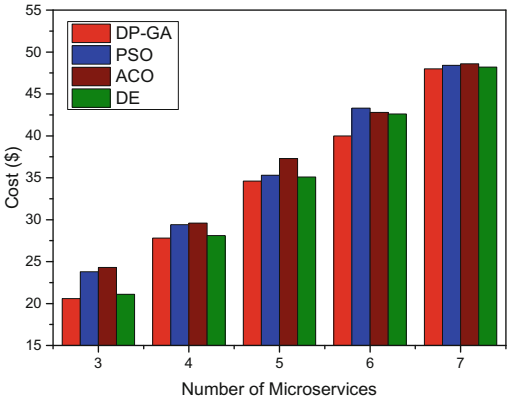

Fig. 3. Comparison of deployment cost with different numbers of microservices.

Next, we compare the effects of four algorithms in deployment cost. The results are shown in Fig. 3. From the information in the figure, it can be seen that DP-GA can always get the least cost among the four algorithms and save about 15% in deployment costs.

Comparison of TRT and Cost Under Different Availability Control Coefficient. We also take the application ca_1 as an example, deploy four microservices, and each microservice has two instances. The availability of the application can be greatly improved through multiple instances. Here, we control the availability in different degrees. Under the same conditions, the deployment schemes obtained by the four algorithms are compared to verify the effectiveness of DP-GA. In order to maximize the application availability, the instances of each microservice are also deployed in different cloud-edge data centers. For

each user center in each deployment scheme, the shortest path with the least time-consuming is selected when the user center requesting applications.

First, we compare average response time, and the experimental results are shown in Fig. 4. We can conclude from this figure that as the availability constraint continues to increase, the location where we deploy microservices will be farther and farther away from the densely populated city center, and the TRT at this time will continue to increase. The TRT is always the smallest in the deployment scheme given by DP-GA, and when k is 0.5, the TRT is reduced by about 30%.

Next, we compare the deployment cost of the four algorithms under different availability control coefficient, as shown in Fig. 5. Comparison results in the figure shows that with the continuous improvement of availability constraints, the deployment location is getting farther and farther away from the city center,

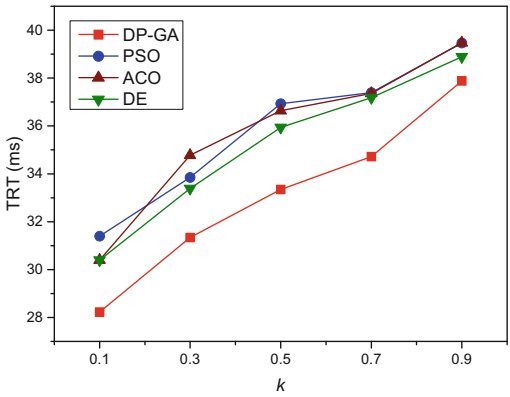

Fig. 4. Comparing average response time with different availability control coefficient.

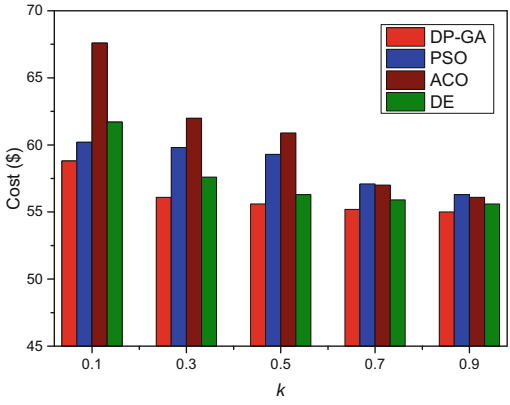

Fig. 5. Comparing deployment cost with different availability control coefficient.

and the cost at this time is constantly decreasing, when k is 0.5, the deployment cost is saved by about 10%. It can be seen that DP-GA is better than other algorithms.

6 Conclusion

In this paper, we studied the problem of optimal application deployment in cloud-edge environments considering availability constraints. To address this problem, we propose a novel genetic algorithm-based method to solve a multi-objective optimization problem with two objectives, that is to minimize the average response time and deployment cost. This method can allocate appropriate virtual machines to microservices, and deploy microservices in appropriate cloud-edge centers to achieve the goal. Based on the real dataset of Shanghai Telecom, we have confirmed through a large number of experimental results that the proposed multi-objective optimization algorithm DP-GA is superior to PSO, ACO and DE.

However, the proposed model and method have certain limitations, and for future work we will adjust and extend our method to adapt to more complex problems. For example, we will further comprehensively consider issues such as privacy, fault tolerance and reliability in the cloud-edge environment to cope with more dynamic environments and more complex user needs.

Acknowledgement. This work is supported by Natural Science Foundation of Hunan Province (No. 2021JJ30278).

References

1. Guerrero, C., Lera, I., Juiz, C.: Genetic algorithm for multi-objective optimization of container allocation in cloud architecture. J. Grid Comput. **16**(1), 113–135 (2018)
2. Guerrero, C., Lera, I., Juiz, C.: Resource optimization of container orchestration: a case study in multi-cloud microservices-based applications. J. Supercomput. **74**(7), 2956–2983 (2018). https://doi.org/10.1007/s11227-018-2345-2
3. Heilig, L., Buyya, R., Voß, S.: Location-aware brokering for consumers in multi-cloud computing environments. J. Netw. Comput. Appl. **95**, 79–93 (2017)
4. Hu, Y., De Laat, C., Zhao, Z.: Multi-objective container deployment on heterogeneous clusters. In: 2019 19th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID), pp. 592–599. IEEE (2019)
5. Li, L., Chen, J., Yan, W.: A particle swarm optimization-based container scheduling algorithm of docker platform. In: Proceedings of the 4th International Conference on Communication and Information Processing, pp. 12–17 (2018)
6. Lin, M., Xi, J., Bai, W., Wu, J.: Ant colony algorithm for multi-objective optimization of container-based microservice scheduling in cloud. IEEE Access **7**, 83088–83100 (2019)
7. Little, J.D., Graves, S.C.: Little's law. In: Chhajed, D., Lowe, T.J. (eds.) Building Intuition, pp. 81–100. Springer, Boston (2008). https://doi.org/10.1007/978-0-387-73699-0_5

8. Liu, B., Li, P., Lin, W., Shu, N., Li, Y., Chang, V.: A new container scheduling algorithm based on multi-objective optimization. Soft. Comput. **22**(23), 7741–7752 (2018). https://doi.org/10.1007/s00500-018-3403-7
9. Lv, L., et al.: Communication-aware container placement and reassignment in large-scale internet data centers. IEEE J. Sel. Areas Commun. **37**(3), 540–555 (2019)
10. Mao, Z., Yang, J., Shang, Y., Liu, C., Chen, J.: A game theory of cloud service deployment. In: 2013 IEEE Ninth World Congress on Services, pp. 436–443. IEEE (2013)
11. Shi, T., Ma, H., Chen, G.: A genetic-based approach to location-aware cloud service brokering in multi-cloud environment. In: 2019 IEEE International Conference on Services Computing (SCC), pp. 146–153. IEEE (2019)
12. Shi, T., Ma, H., Chen, G.: A seeding-based GA for location-aware workflow deployment in multi-cloud environment. In: 2019 IEEE Congress on Evolutionary Computation (CEC), pp. 3364–3371. IEEE (2019)
13. Wen, Z., Cała, J., Watson, P., Romanovsky, A.: Cost effective, reliable and secure workflow deployment over federated clouds. IEEE Trans. Serv. Comput. **10**(6), 929–941 (2016)