Optimized Task Allocation for IoT Application in Mobile Edge Computing

Jialei Liu, Chunhong Liu, Bo Wang, Guowei Gao, Shangguang Wang, *Senior Member, IEEE*

***Abstract*—With the rapid upgrading and explosive growth of Internet of Things (IoT) devices in mobile edge computing, more and more IoT applications with high resource requirements are developed and utilized. Meanwhile, there are large quantities of edge nodes (e.g., switches and edge servers) with limited resources, higher operating costs, and certain failure probabilities in mobile edge computing environment. Therefore, when an IoT application is split into multiple collaborative tasks and offloaded into multiple edge clouds, there is an urgent need to increase the availability level of the task allocation scheme and the resource utilization of edge servers under the condition of certain communication delay. In this paper, we first present a joint optimization objective to evaluate the unavailability level, communication delay, and resource wastage while allocating the same batch of IoT applications to multiple edge clouds. We then propose an approach to minimize the joint optimization objective under the condition of certain communication delay. Finally, we performed a comprehensive simulation experiment analysis to demonstrate that our proposed approach is superior to other related approaches.**

***Index Terms*—mobile edge computing; IoT application; availability; resource wastage; communication delay**

I. INTRODUCTION

WITH the widespread adoption of Internet of Things (IoT) in electronic medical care, disaster response, smart city, intelligent transportation, and smart grid [1], IoT devices such as Raspberry Pi and smartphones are growing in popularity. Cisco forecasted in 2018 that the quantity of IoT devices in the world will increase from 8.6 billion in 2017 to 12.3 billion in 2022, in which more than 422 million IoT devices and connections worldwide will adopt 5G [2]. However, since these IoT devices typically own limited resources, they cannot satisfy the computing requirements of IoT applications [3]. Therefore, mobile edge computing (MEC) emerges as a new computing paradigm that exploits resources near the IoT devices to provide services in a timely manner along with the cloud servers [4]. In MEC, the delay-sensitive and resource-hungry IoT applications from the IoT devices are usually processed in the edge clouds, which consist of some edge nodes (e.g., edge servers, switches, IoT gateways, routers, etc.) with computation, communication, and storage capabilities [5].

Nevertheless, these edge clouds often own the distributed nature and volatile workload, and lack the advanced support systems such as air-conditioning units and power generation equipment that are present in the remote cloud [6]. Therefore, they have very common software and hardware failures (e.g., switch or edge server faults), and then have lower availability than the remote cloud [6]. Besides the increase of unavailability level and service level objective violations, these failures can adversely affect the deployed IoT applications and result in significant performance degradation of the mobile edge computing environment. For example, since the response time of some applications has increased by 500ms, Google experienced system performance degradation which lead to a revenue loss of 20 percent [7]. Moreover, because the existing fault-tolerant mechanisms in the remote cloud typically consume lots of resources, and there are some limited power capacity and small processors in the edge clouds, the existing fault-tolerant mechanisms in the remote cloud are not easy to be applied to the edge clouds [8]. Therefore, how to obtain a high available and resource-efficient task allocation scheme under the condition of certain communication delay is an urgent problem that requires immediate attention [9].

The resource-hungry and delay-sensitive IoT applications are often split onto multiple collaborative tasks that can be independently designed, developed, deployed, and maintained, and then offloaded to the edge servers for processing by multiple containers or virtual machines [10],[11]. Although these edge servers to be selected all work, if there is no availability assessment of allocation scheme of these tasks, some edge servers that are not suitable for running these containers or virtual machines can deteriorate or even go down [12]. Therefore, lack of the availability assessment will further lead to the edge cloud availability with high uncertainty. At present, there are no researches carrying on the availability assessment of task allocation of IoT application, which makes the availability of edge servers very different from the ideal situation. This is, if the availability level of the task allocation scheme is not effectively guaranteed over the long term, the deployment of the IoT application will ultimately fail.

In order to address the above issues, in this paper, we introduce an Optimized Task Allocation Approach (OTAA) of IoT application based on biogeography-based optimization algorithm (BBO) [13] to simultaneously maximize the availability level of task allocation scheme and resource utilization of edge servers under the condition of certain communication delay. To find the equilibrium between the

Jialei Liu, Bo Wang, and Guowei Gao are with the School of Software Engineering, Anyang Normal University, Anyang, China (E-mail: {jialeiliu, bowang, gaoguowei}@aynu.edu.cn).

Chunhong Liu is with the School of Computer and Information Engineering, Henan Normal University, Xinxiang, China (E-mail: lch@htu.edu.cn).

Shangguang Wang is with the State Key Laboratory of Networking and Switching technology, Beijing University of Posts and Telecommunications, Beijing, China (E-mail: sgwang@bupt.edu.cn).

aforementioned objectives, we model the unavailability level, communication delay and the resource wastage, and establish a joint optimization function of them.

Our **main contributions** of this paper include:

- The optimized task allocation problem of the IoT application as described above is formulated, and converted to three contradictory problems which are solved by the unavailability level model, latency model, and resource wastage model.
- On the basis of the above three models, a joint optimization model is first established to measure the unavailability level of task allocation scheme, the communication delay dealing with the IoT applications, and the resource wastage of edge servers. Then, we present the OTAA to minimize the joint optimization objective, i.e., simultaneously maximize the availability level and resource utilization under the condition of certain communication delay.
- We build a mobile edge computing system model, and conduct a comprehensive performance evaluation of our proposed approach. The simulation experiment results demonstrate that our proposed approach can achieve flexible equilibrium among the unavailability level and resource wastage under the condition of certain communication delay.

Organization. Section Ⅱ presents related work; Section Ⅲ proposes our system model; Section Ⅳ introduces our research problem and problem formulation; Section Ⅴ introduces the design and implementation of our proposed approach; Section Ⅵ conducts the simulation experiments; Finally, Section Ⅶ presents conclusions of this paper along with our future work.

II. Related Work

As part of an emerging 5G network, edge cloud has become one of the key enablers for providing the critical IoT services (e.g., content delivery and IoT applications). When these services are offloaded to the edge clouds, the availability level of task allocation scheme of these services, communication delay dealing with the IoT applications, and the resource utilization of edge servers handling these services are pressing issues that need to be improved. At present, there are numerous research works to study them.

For the service availability in multiple edge clouds, Aral et al. [6] exploited the historical failure data to obtain dependencies between failures and model their effect on edge virtual machine availability through a Bayesian networks. Similarly, Soualhia et al. [7] analyzed the data from the edge computing environment via supervised machine learning and statistical techniques to detect and predict the faults at infrastructure-level of edge clouds. Zhu et al. [12] modeled the availability and inter-host network bandwidth cost effect from different placement policy of the mobile edge applications, and the availability level is improved by separating virtual machines of the mobile edge application. Similarly, Yao et al. [14] investigated the equilibrium between maximizing the reliability of virtual machines and minimizing the virtual machine rentals for fog resource provisioning in IoT networks. Furthermore, Maia et al. [15] minimized response time deadline violation, operational cost, and unavailability by investigating how to deploy replicas of applications and requests among these replicas. However, the above research works did not consider the effect of the edge node (e.g., switch and edge server) failure probabilities on the availability level of task allocation scheme of the IoT applications, and the effect of the different task allocation schemes on the resource utilization of edge clouds and the communication delay dealing with the IoT applications.

For the resource utilization of edge clouds, Oueis et al. [16] introduced a customizable algorithm of low complexity small cell clusters establishment and resources management in the case of the consideration of the multi-user computation offloading. Zhao et al. [17] allocated virtual machine replica copies of the IoT application to the edge computing environment via a new framework to minimize the average data traffic. Hu et al. [18] researched the service allocation problem in mobile edge computing to find the equilibrium between load balance and average network delay. Xie et al. [19] proposed an efficient retrieval service and data placement for edge computing to realize the effective control of routing path lengths, forwarding table sizes, and the load balance. Moreover, Pasteris et al. [20] focused on multiple services place problem in a heterogeneous mobile edge computing environment to maximize the total system reward. Farhadi et al. [21] jointly optimized request scheduling and service (data & code) placement to serve time-varying demands under the consideration of system stability and operation cost in a two-time-scale framework. Chen et al. [22] minimized the latency for IoT devices and the monetary cost for Application Service Providers by formulating data-intensive application edge allocation approach. Furthermore, Meng et al. [23] introduced an online algorithm to greedily schedule newly arriving tasks and satisfy the new deadlines by considering whether to take the place of multiple existing tasks. Chen et al. [24] minimized the response time of the data-intensive edge application allocation under storage constraints and load balancing conditions by introducing a data-intensive application allocation policy with genetic algorithm. Khan et al. [25] introduced a mathematical model to calculate the overall computational time and energy consumption of mobile cloud application models. Guo et al. [26] carried out the assignment of tasks in an online fashion to realize an optimal power-delay equilibrium in the system by designing policies. Maia et al. [27] minimized the potential violation of their QoS requirements by jointly investigating the load distribution and placement of scalable IoT applications. Goudarzi et al. [28] minimized the execution time and energy consumption of IoT applications by exploiting a new application placement technique with the memetic algorithm to realize batch application placement. Peng et al. [29] proposed an end-edge-cloud collaborative computing offloading approach based on improved Strength Pareto Evolutionary algorithm to minimize simultaneously time consumption and energy consumption of mobile users, and resource utilization of edge servers. Cheng et al. [30] proposed three algorithms to reduce the latency and energy consumption in data shared mobile edge computing systems by studying the task assignment algorithm. Although these research works have studied various types of resource consumption and latency or response time in edge computing environment, they have not considered the balance between resource wastage of edge servers and availability level of task allocation scheme of IoT

application under the condition of certain communication delay.

Different from the above study, we firstly model the unavailability level of task allocation scheme, the communication delay dealing with the IoT applications, and the resource wastage of edge servers in multiple edge clouds with fiber backhaul network. Then, we propose the joint optimization problem and solve it by the OTAA under the condition of the contradictory relation of resource utilization, communication delay and availability level. For ease of understanding, Table Ⅰ lists the key notations in this paper.

TABLE Ⅰ. KEY NOTATIONS

Notation	Description
V	the set of edge clouds including heterogeneous quantity of edge servers
$\mathcal{E}$	the set of network links between the edge clouds
M	the total quantity of edge clouds
ξ	the set of serving edge clouds, $\xi \subset V$
$\mathcal{H}$	the total quantity of serving edge clouds
$\mathcal{F}$	the set of IoT devices
$\mathcal{K}$	the total quantity of IoT devices
N	the number of heterogeneous edge servers
Q	the number of heterogeneous containers or virtual machines
N_m	the set of edge servers in the edge cloud m
$V_{m,n}$	the set of the containers or virtual machines in the *n-th* edge server of edge cloud m.
n_m	the *n-th* edge server of edge cloud m where $n_m \epsilon N_m$
Z_i	the quantity of tasks in the IoT application i
L	the amount of IoT applications offloaded at some point.
φ	a batch of IoT applications
$N_{i,m}$	the number of available containers or virtual machines in the cluster i
$P_{i,m}$	the failure probability of the switch inside the edge cloud m associated with the cluster i
E_i	the mathematical expectation of the quantity of available containers or virtual machines in the cluster i
σ_i	the standard deviation of the number of available containers or virtual machines in the cluster i
T	the threshold value of communication delay.
Φ	joint optimization objective
P	the size of the population or the number of islands
S^*	the maximum number of species in an island
P_s	the probability that the island X includes exactly s species

III. SYSTEM MODEL

In this section, we will propose our system model of the mobile edge computing environment.

A. System Description

As shown in Fig. 1, we build a mobile edge computing system including multiple edge clouds (ECs) connected together via fiber backhaul network using full mesh topology [31], [32]. Each edge cloud is endowed with computational and storage capacities by deploying some heterogeneous edge servers interconnected using switches, and accessed via a small cell base station covering a specified area that can receive and forward offloading requests from IoT devices. Each application service providers can rent the small cell base stations from communication facility providers to deploy IoT applications, in which each IoT application consists of a set of indivisible tasks to be executed in some edge clouds. As we all know that switch failures take up most of the downtime in remote cloud data centers [33], and thus each edge cloud is called an individual fault domain in our system.

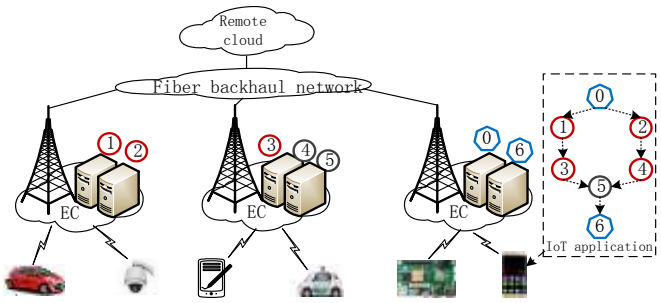


Fig. 1. Mobile edge computing system

B. Network Model

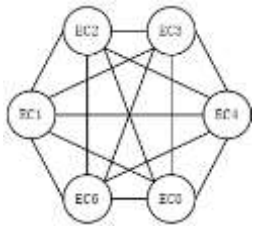


Fig. 2. Mobile edge computing network topology

As shown in Fig.2, the mobile edge computing network topology is modeled as an undirected graph denoted by G = (V,$\mathcal{E}$), where the vertices V and the edges $\mathcal{E}$ represent the set of edge clouds including heterogeneous quantity of edge servers and network links between the edge clouds, respectively. Since the tall buildings in the city produce great interference to the wireless signal, the vertices of the graph are interconnected through fiber backhaul, and the propagation latency between the edge clouds is assumed to be load independent. Without loss of generality, we take into account a set of edge clouds V={1,...,M} where M is the total quantity of edge clouds, a set of serving edge clouds ξ ={1,...$\mathcal{H}$} and $\xi \subset V$ where $\mathcal{H}$($\mathcal{H} \leq M$) is the total quantity of serving edge clouds, and a set of IoT devices $\mathcal{F}$ ={1,...,$\mathcal{K}$} where $\mathcal{K}$ is the total quantity of IoT devices. Each IoT device $i \in \mathcal{F}$ has a corresponding mobile user and is associated with an edge cloud $m \in \xi$ to which offloading tasks are sent.

C. Edge Server Model

The edge servers are often deployed inside the edge clouds, and then the tasks of the IoT applications can be offloaded by nearby IoT devices via one wireless hop. There are N heterogeneous edge servers and Q heterogeneous containers or virtual machines distributed in different edge clouds of the mobile edge computing environment in total, and the quantity of the edge servers in set N_m and the containers or virtual machines in set $V_{m,n}$ is different in different edge clouds and different edge servers, respectively, where N_m is the set of edge servers in the edge cloud m; n_m is the *n-th* edge server of edge cloud m where $n_m \epsilon N_m$; $V_{m,n}$ is the set of the containers or virtual machines in the *n-th* edge server of edge cloud m. Each edge server owns various types of resources such as CPU, memory, and bandwidth, etc. Furthermore, the three resource dimensions

described above can be represented by the set Λ = {CPU, mem, bw}. Meanwhile, since there are limited amount of the computation, memory and bandwidth resources in each edge server, only a limited quantity of the IoT applications can be processed in it.

D. Application model

Consider that an IoT application consists of a set of collaborative tasks $\{t_0,...,t_{Z_i-1}\}$ to be executed in some containers or virtual machines of multiple edge clouds where Z_i denotes the quantity of tasks in the IoT application *i*. The dependencies of these tasks can be denoted by a directed acyclic graph (DAG), in which a directed link (t_a,t_b) indicates that the task t_b must be processed after the task t_a. The tasks with 0 in-degree and 0 out-degree are called the entry task and the exit task (e.g., t_0 and t_6 in Fig. 1), respectively. As a reminder, each entry task passes some initial data to the IoT application, and the outputs of each exit task are combined to be the computation result of the IoT application. When an IoT application is generated on an IoT device with a deadline, it will be split into some collaborative tasks and offloaded to multiple nearby edge clouds, and then transferred to some edge servers according to the propagation latency between the edge clouds. Since IoT devices appear in the mobile edge computing environment and generate the IoT applications in arbitrary order and time, a batch of IoT applications can be denoted by the set φ ={1,...,*L*} where L ($L \le \mathcal{K}$) is the total amount of IoT applications to be offloaded at some point.

IV. Problem Statement and Formulation

In this section, we first introduce our problem statement, and then propose availability model, resource wastage model of the IoT application, and communication delay model, and finally introduce an optimization formulation to simultaneously minimize the overall resource wastage and unavailability level of the task allocation scheme under the condition of certain communication delay.

A. Problem Statement

The rapid refreshment of IoT devices and the explosion of their number have result in a sharp increase in the quantity of resource-hungry and delay-sensitive IoT applications. Although the resource configuration (e.g., computing, storage, bandwidth or battery capacity) of these IoT devices has been greatly improved, they still cannot meet the computing requirements of these IoT applications. Therefore, an IoT application needs to be split into multiple collaborative tasks, which are offloaded onto the edge clouds and processed by a cluster of containers or virtual machines on the edge servers. Moreover, it makes the IoT devices to reduce power consumption and speed up the calculation process, and then makes it possible to run emerging IoT applications on IoT devices. However, consider that edge server resources (such as CPU, memory, and bandwidth) are generally more expensive than those in the remote cloud, and then edge node (e.g., switch and edge server) failures are quite common at edge clouds. One major challenge is how to resource-efficiently, delay-sensitively, reliably allocate the limited resources from edge servers to these IoT applications under the consideration of heterogeneous edge node failure probabilities.

B. Availability Model

Since edge server and switch failures are common, and own the weak correlations and heterogeneity, for the sake of clarity, we consider a simple scenario that there is a switch failure and an edge server failure in multiple edge clouds. In this case, the edge server failure typically results in all containers or virtual machines on it to fail, and the switch failure typically results in all edge servers connecting it to be inaccessible. Meanwhile, when an IoT application of the set φ is offloaded to some edge clouds, multiple containers or virtual machines with different sizes will be requested to process its tasks. Therefore, it is very necessary to propose a probability model of the number of available containers or virtual machines to reduce the unavailability level of the task allocation scheme of the IoT application.

For a given task allocation scheme of IoT application *i*, these tasks are processed by a cluster including Z_i containers or virtual machines, and encounter two failure events, i.e., edge cloud *m with* switch failure along with no edge server failure (i.e., event *E1*) and concurrently with an edge server failure (i.e., event *E2*). The probability of event *E1*, *E2* and *E* can be calculated by the formulation (1), (2) and (3).

$$P(E1) = C_N^0 P_k^0 \prod_{l=1}^{N} (1-P_l) \tag{1}$$

$$P(E2) = C_N^1 P_k^1 \prod_{l=1, l\neq k}^{N} (1-P_l) \tag{2}$$

$$P(E) = P(E1) + P(E2) \tag{3}$$

where P_l and P_k represent the failure probability of different edge servers; N can be obtained by $M \cdot |N_m|$.

The number of unavailable containers or virtual machines in this cluster due to the event *E1* where edge cloud *m* with failure switch is represented by $\delta_{i,m}^{E1}$, which can be calculated by the formulation (4).

$$\delta_{i,m}^{E1} = \sum\nolimits_{n_m} \sum\nolimits_j X_{i,j}^{m,n_m} P(E1 \mid E) = \sum\nolimits_{n_m} \sum\nolimits_j X_{i,j}^{m,n_m} \frac{P(E1)}{P(E)} \tag{4}$$

where the allocation of a container or virtual machine in this cluster is represented by $X_{i,j}^{m,n_m}$; if the container or virtual machine $j \epsilon V_{m,n}$ of the cluster *i* is assigned to edge server n_m belonging to edge cloud *m*, then $X_{i,j}^{m,n_m} = 1$; otherwise, $X_{i,j}^{m,n_m} = 0$.

To obtain the value of $\delta_{i,m}^{E2}$, two cases are considered: In one case, the edge cloud *m* with failure switch contains the failed edge server, and in the other case, the failed edge server is not placed in the edge cloud *m* with failure switch. $\delta_{i,m}^{E2}$ is computed by adding the sum of containers or virtual machines accommodated in the edge cloud *m* with failure switch to the expected number of unavailable containers or virtual machines due to an edge server failure outside the edge cloud *m*, as shown in formulation (5).

$$\delta_{i,m}^{E2}=\left(\sum_{n_m}\sum_{j}X_{i,j}^{m,n_m}+\frac{M-1}{M}\sum_{m}\sum_{n_m}\sum_{j}X_{i,j}^{m,n_m}P_k\right)\frac{P(E2)}{P(E)} \quad (5)$$

When the switch of the edge cloud m fails, the number of available containers or virtual machines in the cluster i can be represented by $N_{i,m}$, which can be computed by the formulation (6).

$$N_{i,m}=Z_i-\delta_{i,m}^{E1}-\delta_{i,m}^{E2} \quad (6)$$

The mathematical expectation E_i of the quantity of available containers or virtual machines in the cluster i can be calculated by subtracting the expected quantity of unavailable containers or virtual machines from the total quantity of containers or virtual machines in the cluster i, as shown in the formulation (7).

$$E_i=Z_i-\sum_{m}P_{i,m}\delta_{i,m}^{E1}-\sum_{m}P_{i,m}\delta_{i,m}^{E2} \quad (7)$$

where $P_{i,m}$ represents the failure probability of the switch inside the edge cloud m associated with the cluster i;

Therefore, according to the definition of standard deviation, the standard deviation σ_i of the number of available containers or virtual machines in the cluster i is represented as the formulation (8).

$$\sigma_i=\sqrt{\frac{1}{M-1}\sum_{m}(N_{i,m}-E_i)^2} \quad (8)$$

Finally, considering that the actual number of available containers or virtual machines in this cluster changes from S_i-σ_i to $S_{i+}\sigma_i$, if the standard deviation σ_i is larger, the number of available containers or virtual machines fluctuates significantly. This is, the task allocation scheme of IoT application i has higher unavailability, which increases the risk of the task allocation failure. Therefore, the normalized standard deviation of the quantity of available containers or virtual machines is exploited to represent the unavailability level of the task allocation scheme of IoT application i, as shown in the formulation (9).

$$unavailability_i=\frac{\sigma_i}{E_i} \quad (9)$$

C. Resource Wastage Model

We propose a resource wastage model based on [34], [35] to quantify the resource wastage of all dimensions and balance the residual resources of the λ-th edge server along different dimensions. The resource utilization of the λ-th edge server is expressed as the total amount which is occupied by all containers or virtual machines hosted on the edge server. One resource that is 100% used can cause severe performance degradation and trigger a real-time container or virtual machine migration that requires additional CPU processing time of the migration node [36]. Therefore, all dimension resource utilization of the λ-th edge server should be set to an upper bound capped at 90%, and then the resource wastage $resW_\lambda$ of the λ-th edge server can be calculated by the formulation (10).

$$resW_\lambda=\frac{\eta_\lambda}{D}\times\sum_{\alpha\neq\beta}\sum_{\beta=1}^{D}\frac{\left|\left(T_\lambda^\alpha-\sum_{\delta=1}^{Q}\left(b_{\delta\lambda}R_\delta^\alpha\right)\right)-\left(T_\lambda^\beta-\sum_{\delta=1}^{Q}\left(b_{\delta\lambda}R_\delta^\beta\right)\right)\right|}{\sum_{\delta=1}^{Q}\left(b_{\delta\lambda}R_\delta^\alpha\right)+\sum_{\delta=1}^{Q}\left(b_{\delta\lambda}R_\delta^\beta\right)} \quad (10)$$

where the binary variable η_λ represents whether the λ-th edge server is in use (value 1) or not (value 0); D represents the total quantity of resource type in the set Λ; α (or β) represents a kind of the resource type in the set Λ; R_δ^α and R_δ^β are the demands for resource type α and β on the δ-th container or virtual machine of the set $V_{m,n}$, respectively. T_λ^α and T_λ^β represent the utilization threshold of resource type α and β in the λ-th edge server, respectively. The binary variable $b_{\delta\lambda}$ indicates whether δ-th container or virtual machine is allocated to the λ-th edge server or not, that is, if the δ-th container or virtual machine is allocated to the λ-th edge server, then $b_{\delta\lambda}$=1, otherwise $b_{\delta\lambda}$=0.

D. Latency Model

When the collaborative tasks of the IoT application i are offloaded to the edge clouds, we assume that these tasks are first pre-assigned randomly to the containers or virtual machines, and then allocated initially the containers or virtual machines handling these tasks to the edge servers. If the containers or virtual machines handling these tasks are placed on different edge servers, these containers or virtual machines need to communicate with each other, and increase the communication delay of the IoT application i. Since the edge clouds communicate with each other through optical fiber, the propagation latency between the edge clouds is assumed to be load independent. The communication delay between containers or virtual machines is mainly determined by the bandwidth and the amount of data transferred of the sending containers or virtual machines. For the sake of clarity, we consider a simple scenario that the IoT devices offload all tasks to the edge servers near an edge cloud within a certain period of time, and the communication delay from the IoT device to the edge cloud can be considered as no change, this is due to that the amount of data transmitted between the edge cloud and the IoT device is a certain amount. Furthermore, we build a latency model (as shown in formulation (11)) based on an IoT application model including three tasks, as shown in Fig. 3, in which two tasks are sending tasks, and the third task is receiving task and can be processed after the results of all sending tasks are transmitted to the third task. The latency model represents the sum of the communication time taken by two sending tasks, which transfer data to the third task through the bandwidth. Although the result of the latency model is not all of the application time, it has the similar variation pattern as the application completion time. Therefore, we only need to consider the communication delay of the containers or virtual machines handling the tasks in the IoT application i, which are allocated to the different edge servers, as shown in the formulation (11).

$$Latency_i=\sum_{q=1}^{Z_i}x_{i,q}\frac{data_q}{bw_q} \quad (11)$$

where bw_q and $data_q$ denote the data bandwidth and the amount of data sent by the q-th container or virtual machine dealing with the IoT application i, respectively; $x_{i,q}$ indicates whether the container or virtual machine hosting the q-th task of the IoT application i is a sender, such that $x_{i,q}=1$ if the q-th container or virtual machine is a sender; otherwise, $x_{i,q}=0$.

E. Problem Formulation

When IoT devices appear online in a given mobile edge computing system and produce a batch of IoT applications at some point, each IoT application consists of multiple collaborative tasks, which are pre-assigned randomly to the containers or virtual machines. These containers or virtual machines handling these tasks need to be allocated initially to multiple edge servers of the edge clouds by the exploitation of one allocation strategy. If these containers or virtual machines are allocated to different edge clouds, although the availability level is improved, their communication delay and resource wastage may be increased to varying degrees. Conversely, the more densely they are allocated, the less their resource wastage and communication delay, and the lower their availability level. That is, if these containers or virtual machines are on the same edge server, then they do not occupy the bandwidth resources of the edge server, thus there will be very little communication delay between them. Instead, if they are not on the same edge servers, then they need to exploit some edge server bandwidth resources to transfer data, thus increasing the communication delay. Similarly, if these containers or virtual machines are on different edge servers, then, these edge servers all need to be started, thus wasting a large number of resources. In contrast, if these containers or virtual machines are distributed across only a few edge servers centrally, the resources of those edge servers are fully utilized. Furthermore, how to allocate initially these containers or virtual machines to the edge clouds has significant impact on the resource wastage level of edge computing environment, availability level of the task allocation scheme, and the communication delay of the tasks in all applications. Therefore, we need to find the optimized allocation scheme of these IoT applications to simultaneously minimize the unavailability level and resource wastage of these edge clouds in satisfaction of the limited resource capacity and the certain communication delay while processing L IoT applications. More specifically, we adopt a joint optimization objective Φ to measure the joint optimization problem as shown in the formulation (12).

$$\Phi = \theta_1 \sum_{i=1}^{L} unavailability_i + \theta_2 \sum_{\lambda=1}^{N} resW_\lambda + \theta_3 \sum_{i=1}^{L} Latency_i \quad (12)$$

s.t.

$$0 < \theta_1, \theta_2, \theta_3 < 1, \text{ and } \theta_1 + \theta_2 + \theta_3 = 1 \quad (13)$$

$$\sum_{i=1}^{L} Latency_i < T \quad (14)$$

$$\sum_{i=1}^{L} Z_i \le Q \quad (15)$$

$$\max_{1 \le j \le Q} \left(\sum_{t=1}^{\sum_{i=1}^{L} Z_i} x_{tj} \right) = 1, x_{tj} = 0 \text{ or } 1 \quad (16)$$

$$\sum_{j=1}^{\sum_{i=1}^{L} Z_i} D_j^{CPU} y_{jz} < C_z^{CPU} \quad (17)$$

$$\sum_{j=1}^{\sum_{i=1}^{L} Z_i} D_j^{mem} y_{jz} < C_z^{mem} \quad (18)$$

$$\sum_{j=1}^{\sum_{i=1}^{L} Z_i} D_j^{bw} y_{jz} < C_z^{bw} \quad (19)$$

$$\sum_{z=1}^{N} y_{jz} = 1, y_{jz} = 0 \text{ or } 1 \quad (20)$$

where θ_1, θ_2, and θ_3 denote the tunable positive weights; T denotes the threshold value of communication delay; Formulation (14) shows that the total communication delay for all sending tasks of L IoT applications is less than the threshold value T; Formulations (15) shows that the total number of tasks of L IoT applications is less than the total number of containers or virtual machines; Formulation (16) shows that each container or virtual machine can only handle any of these tasks, such that $x_{tj} = 1$ if the t-th task is handled by the j-th container or virtual machine; otherwise, $x_{tj} = 0$; Formulations (17) to (19) show that the total resource demand of the containers or virtual machines handling these tasks on the z-th edge server is less than the idle resource capacity of the edge server; Formulation (20) shows that a container or virtual machine can only be assigned to an edge server, such that $y_{jz} = 1$ if the j-th container or virtual machine is handled by the z-th edge server; otherwise, $y_{jz} = 0$.

V. Approach Design

Since a series of optimization problems are solved by the BBO algorithm, which has been proved to be one of the fastest-growing biology-based algorithms [37], it is adopted for the task allocation of the IoT application. In this section, we firstly present the BBO algorithm, then propose our improvement scheme including the mapping model and the definition of operators, and finally present the scheme of design and implementation of the OTAA.

A. BBO Algorithm

An archipelago (i.e., ecosystem) including multiple islands (i.e., habitats or individuals) represents the population of candidate solutions in the BBO algorithm [13]. The Habitat Suitability Index (HSI) is affected by the Suitability Index Variables (SIVs) (e.g., rainfall, temperature, etc.), and denotes the fitness of a candidate solution. Therefore, a vector of SIVs denotes a candidate solution. There are two key operators, i.e., migration and mutation in the BBO algorithm, in which migration operator is a significant feature that distinguishes it from other population-based optimization algorithms, and it also improves the quality of low-HSI solutions by probabilistically sharing SIVs among candidate solutions; and then some SIVs in a candidate solution are probabilistically replaced with randomly generated new SIVs by the mutation operator.

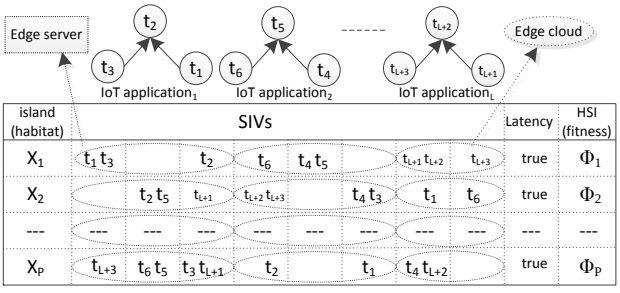


Fig. 3. The mapping model of BBO algorithm

B. BBO-based Optimized Task Allocation

To adopt the BBO algorithm, a mapping model is proposed to map the optimized task allocation problem of the IoT application to an ecosystem (i.e., population), as shown in Fig. 3. P denotes the size of the population; Latency indicates whether the communication delay processing L IoT

applications is within the time given, such that the value of Latency is true if it is within the time given; otherwise, the value of Latency is false; Φ_1, Φ_2, …, Φ_P denote the fitness of the candidate solutions X_1, X_2,…, X_P, respectively; each SIV denotes an edge server to which the collaborative tasks of the IoT applications are assigned, and is represented by a dotted rectangle; each dotted ellipse denotes an edge cloud including some SIVs. All candidate solutions have the same joint optimization objective (i.e., formulation (12)) and resource and latency constraints. That is, each candidate solution must process the L IoT applications in certain communication delay, and optimize itself by sharing information with other candidate solutions to optimize the whole population.

Considering the above mapping model and the specific features of the optimized task allocation problem of IoT application, we then redefine the parameters and operators of the BBO algorithm.

Definition 1 (Migration operator).The migration operator is denoted symbolically by $\mathcal{M}(\bullet)$, which is a probabilistic operator that modifies island X according to its emigration rate μ_s and immigration rate λ_s (as shown in formulation (21)) [13],[38]. The formulation (22) denotes the migration operation of the ecosystem from island X_k to island X_j.

$$\begin{cases} \lambda_s = I \cdot \left(1 - s/S^*\right) \\ \mu_s = E \cdot s/S^* \end{cases} \tag{21}$$

$$\mathcal{M}\left(X_{j,\tau}, X_{k,\rho}\right) = \begin{cases} X_{k,\rho}, \text{ if } \lambda_j > \mathrm{R}_1 \text{ and } \mu_k > \mathrm{R}_2 \\ X_{j,\tau}, \text{ otherwise} \end{cases} \tag{22}$$

where S^* denotes the maximum number of species in an island; E and I represent the maximum emigration and the maximum immigration rate, respectively; X_j and X_k represent the *j-th* and *k-th* island of ecosystem, respectively; λ_j and μ_k denote the immigration rate and emigration rate of the *j-th* and *k-th* island, respectively; $X_{j,\tau}$ denotes the $\tau - th$ SIV of the habitat X_j; $X_{k,\rho}$ denotes the $\rho - th$ SIV of the habitat X_k; R_1 and R_2 denote the one-off random numbers in (0,1).

Definition 2 (Mutation operator). The mutation operator is represented symbolically by $U(\bullet)$, which is a probabilistic operator that modifies SIVs of island according to a mutation probability m_s. The formulation (25) shows that the mutation operation of the τ-*th* SIV in the island X_s.

$$P_s = \begin{cases} \dfrac{1}{1 + \sum_{d=1}^{S^*} \dfrac{\lambda_0 \lambda_1 \ldots \lambda_{d-1}}{\mu_1 \mu_2 \ldots \mu_d}}, & \mathrm{s} = 0 \\ \dfrac{\lambda_0 \lambda_1 \ldots \lambda_{s-1}}{\mu_1 \mu_2 \ldots \mu_s \left(1 + \sum_{d=1}^{S^*} \dfrac{\lambda_0 \lambda_1 \ldots \lambda_{d-1}}{\mu_1 \mu_2 \ldots \mu_d}\right)}, & 1 \le s \le S^* \end{cases} \tag{23}$$

$$m_s = m^* \cdot \left(1 - \frac{P_s}{P^*}\right) \tag{24}$$

$$U(X_{s,\tau}) = \begin{cases} \tilde{X}_{s,\tau}, \text{ if } \mathrm{m_s} > R_3 \\ X_{s,\tau}, \text{ otherwise} \end{cases} \tag{25}$$

where P_s denotes the probability that the island X includes exactly s species, as shown in formulation (23) [39]; m_s denotes the mutation probability of the island X, as shown in formulation (24); P^* and m^* represent the maximum value of the probability P_s and the mutation probability m_s, respectively; $\tilde{X}_{s,\tau}$ denotes a new SIV; R_3 denotes the one-off random numbers in (0,1).

Definition 3 (Removal operator). The removal operator is denoted symbolically by $\mathcal{R}(\bullet)$, which identify the overloaded edge servers of each island and replace them with other edge servers in satisfaction of the limited resource capacity and the certain communication delay (i.e., $\sum_{i=1}^{L} Latency_i < \mathrm{T}$). The formulation (26) shows that the removal operator generates a new island $\tilde{X}$ by adjusting the island X under the above constraints.

$$\mathcal{R}(X) = \tilde{X}, \ \textit{if} \ \sum_{i=1}^{L} Latency_i < T \tag{26}$$

Definition 4 (Elitism operator). The elitism operator is represented symbolically by $\tilde{E}(\bullet)$, which ensures that the best e islands are not lost from one generation to the next. The formulation (27) shows that the best e islands at the beginning of each generation are saved into a set $\{X_{P-e+1}, ..., X_P\}$, and then replace the worst e islands of the new population set $\tilde{X}$ with the set at the end of the generation, while the e islands satisfy the resource capacity and the certain communication delay (i.e., $\sum_{i=1}^{L} Latency_i < T$).

$$\tilde{E}(\tilde{X}) = \{\tilde{X}_1, \tilde{X}_2, ..., \tilde{X}_{P-e}\} \cup \{X_{P-e+1}, ..., X_P\} \tag{27}$$

C. Scheme of Design and Implementation of the OTAA

In this section, we propose the scheme of design and implementation of the OTAA with the BBO algorithm to simultaneously minimize the unavailability level of task allocation scheme and resource wastage of edge servers in satisfaction of the limited resource capacity and the certain communication delay. The pseudocode of the OTAA is presented in Algorithm 1.

Algorithm 1: Optimized Task Allocation Approach (OTAA)

Input: All parameters of the IoT applications and the MEC

Output: the optimized task allocation scheme

1 Initialize all parameters of BBO algorithm
2 Initialize P islands randomly
3 Initialize λ_s, μ_s, m_s by the formulation (21), (23), (24)
4 Order all islands by the Φ
5 **for** count=1 to G **do**
6 Save the e elite islands
7 Migrate the non-elite island $X_{j,\tau} \leftarrow \mathcal{M}\left(X_{j,\tau}, X_{k,\rho}\right)$
8 Mutate the non-elite island $X_{s,\tau} \leftarrow U(X_{s,\tau})$
9 Remove the overloaded edge servers $X \leftarrow \mathcal{R}(X)$
10 Order all islands by the Φ recomputed.
11 Replace the e islands with the elites $\tilde{X} \leftarrow \tilde{E}(\tilde{X})$
12 Reorder all islands by the Φ
13 **end for**
14 **return** the optimized task allocation scheme

All parameters of the OTAA are first initialized in the Algorithm 1, and then *P* islands in the ecosystem are randomly initialized in satisfaction of the limited resource capacity in lines 1-3. Second, it orders these islands by the formulation (12). Third, its optimization loop begins, and then saves the *e* elite solutions in line 6. Fourth, each non-elite island is probabilistically mutated and modified by the mutation operator (i.e., ***Definition 2***) and migration operator (i.e., ***Definition 1***), respectively, and then the overloaded edge servers in each island are removed by the removal operator (i.e., ***Definition 3***).Finally, the Φ of each island is recalculated to all islands of the ecosystem, the worst *e* islands of the new population are replaced with *e* elites using the elitism operator (i.e., ***Definition 4***), and then jumps to the step 3 for next iteration after reordering all islands. The cycle ends at a certain number of times (i.e., *G*).

VI. Performance Evaluation

In this section, we build a simulation experiment environment to evaluate the performance and effectiveness of the OTAA for the IoT applications.

A. Experiment Setup

In our simulation, there are 25 edge clouds equally distributed in a 5G network scenario with full mess topology based on our extended CloudSim simulator [40],[41]. In this scenario, each edge cloud consists of a base station interconnected with other base stations via fiber backhaul network, a certain number of heterogeneous edge servers jointly connected via a switch, and multiple IoT devices via wireless access network. Notice that the number of edge servers in each edge cloud randomly selects from the integer set [4, 6]. The configuration parameter of each edge server randomly selects from the set {HP ProLiant G4 (i.e., 4GB of RAM, 3720 MIPS, 1TB of storage, and 1GB/s network bandwidth), HP ProLiant G5 (i.e., 4GB of RAM, 5320 MIPS, 1TB of storage, and 1GB/s network bandwidth)}[42], and randomly deployed into multiple edge clouds. Considering that switches and edge servers own heterogeneous failure probabilities, their value is set randomly between 0.05 ~ 0.15 and 0.02 ~ 0.12 [33], respectively. When IoT devices appear online and produce a batch of IoT applications at some point, each IoT application consists of three collaborative tasks, which are pre-assigned randomly to some heterogeneous containers or virtual machines. Please note that each container or virtual machine can only handle any of these tasks. Since the experimental results under the condition of virtual machine case also apply to the container case, we only need to discuss the topic of virtual machines after this section. The bandwidth requirement of each virtual machine randomly selects from the set [10, 50] Mbps. The amount of data sent by each virtual machine dealing with the IoT application is set randomly in [1, 2] Mb. Its CPU and memory requirement randomly select from the set {2000 MIPS and 3.75 GB, 500 MIPS and 0.6 GB, 1000 MIPS and 1.7 GB, 2500 MIPS and 0.85 GB}[42]. Its disk requirement is set at 1 GB. The population size and generation number *G* are set at 100 and 300, respectively. *E* and *I* are both assigned to 1, the elite number *e* and the maximum mutation m^* are set at 2 and 0.1, respectively [39], [43]. Moreover, the tunable positive weights $\theta_1,\theta_2,\theta_3$ are all set at 1/3; the threshold value of communication delay *T* is set at 2s.

To evaluate the performance and effectiveness of the OTAA, we compare the OTAA with the following benchmark approaches.

- ***Random Allocation (RA)***: Randomly selects the edge server to host each virtual machine when there are multiple edge server candidates that satisfy the constraints.
- ***First Fit (FF):*** Selects the edge server that meets the resource requirements first to host each virtual machine when there are multiple edge server candidates that satisfy the constraints.
- ***Particle Swarm Optimization (PSO):*** Selects the edge server to host each virtual machine based on particle swarm optimization algorithm when there are multiple edge server candidates that satisfy the constraints [44].

B. Experimental Results and Evaluation

Next, the performance and effectiveness of the OTAA are first compared with the other related approaches in terms of unavailability level, resource wastage, and communication delay while processing a batch of IoT applications. The effect of experiment parameters including the failure ranges of the edge server and switch, the number of the base stations, and the number of IoT applications are analyzed in the rest of this section.

1) Comparison of Optimization Objectives

The first group experiment is to compare the OTAA with the other three approaches to evaluate its performance in terms of unavailability level, resource wastage, and communication delay while processing a batch of IoT applications. In this section, the number of IoT applications, base stations, edge servers and virtual machines were set at 20, 25, 127 and 60, respectively; the tunable positive weights $\theta_1,\theta_2,\theta_3$ were all set at 1/3; the failure ranges of the edge server and switch were set randomly within 0.02 ~ 0.12 and 0.05 ~ 0.15, respectively; the threshold value of communication delay *T* was set at 2s.

As shown in Figs. 4 to 6, the average unavailability level of FF is the highest of all approaches. This is due to that FF makes it very easy to allocate the virtual machines processing an IoT application to the same edge server or base station, and then its average resource wastage is not the highest. The reason for the lower average unavailability level of RA is that these edge servers are randomly selected, and the probability of being in the same base station is relatively small. Therefore, its average resource wastage is the highest of all approaches; instead, its average unavailability level is not too high. Since PSO allocates each virtual machine to an edge server that provides the least increase of the joint optimization objective, its average unavailability level, average resource wastage, average communication delay are both lower than RA and FF, but it's still higher than OTAA. Although PSO is a heuristic approach with good performance, PSO is more likely to cluster in similar groups, while the OTAA exploits a new stochastic evolutionary algorithm (i.e., BBO) to look for global optimization, its solution does not necessarily have an inherent clustering trend. Therefore, the average unavailability level for using the OTAA is 15%, 52% and 10% less than the three approaches, respectively; the average resource wastage for using the OTAA

is 199%, 8%, and 6% less than the three approaches, respectively; the average communication delay for using the OTAA is 44%, 33%, and 13% less than the three approaches, respectively.

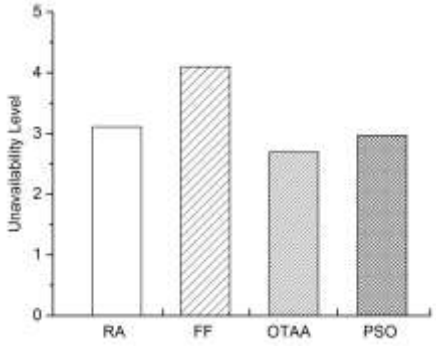


Fig. 4. Comparison of the unavailability level

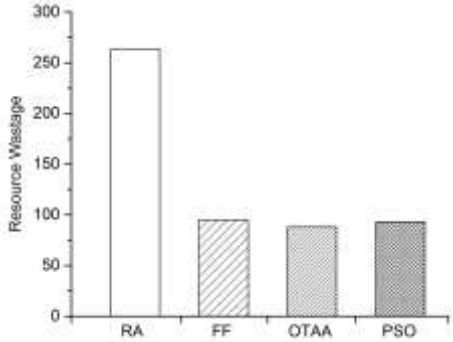


Fig. 5. Comparison of the resource wastage

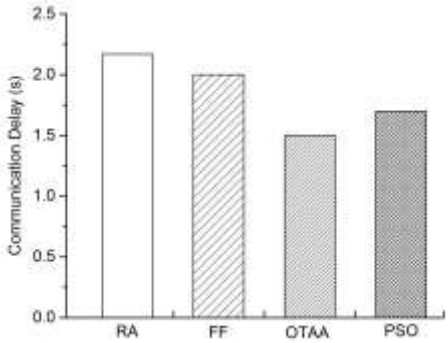


Fig. 6. Comparison of the communication delay

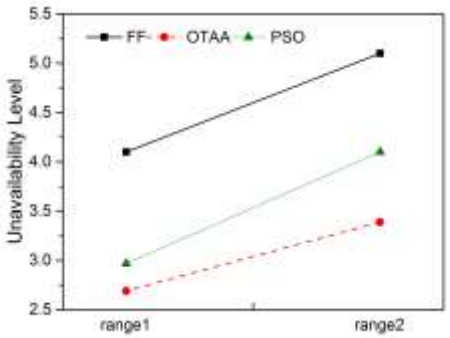


Fig. 7. The effect of different failure ranges of the edge server and switch. Each edge server and each switch can be assigned to a fault range (i.e., range1 and range2), respectively; the average unavailability level of all approaches as the size of the failure range increases.

In next section, we further analyze the impact of the failure ranges of the edge server and switch, the number of the base stations, and the number of the IoT applications on the unavailability level, resource wastage, and communication delay of the FF, OTAA, and PSO (as shown in Figs. 7 to 9). Please note that since the average communication delay (i.e., 2.17s) of RA exceeded the threshold value T of communication delay, RA will not be discussed below.

2) Effect of the Failure Ranges of Edge Server and Switch

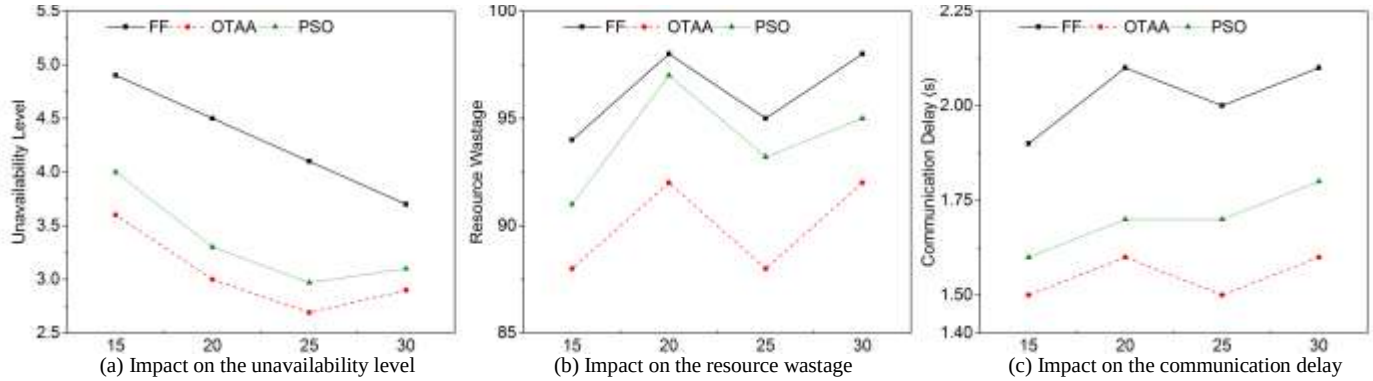
Fig. 7 shows the impact of the failure ranges of the edge server and switch on the unavailability level. To show this effect more clearly, the number of IoT applications, base stations, edge servers and virtual machines was set at 20, 25, 127 and 60, respectively; the tunable positive weights $\theta_1,\theta_2,\theta_3$ were all set at 1/3. With the increase of the failure ranges of the edge server and switch (i.e., from range1 (i.e., [0.02, 0.12] and [0.05, 0.15]) to range2 (i.e., [0.02, 0.22] and [0.05, 0.25]), the average unavailability level of each approach tends to increase overall. That is due to that the quantity of high failure probabilities for the edge servers and switches in range2 are more than the range1. Therefore, when some IoT applications is offloaded to the edge clouds with the higher failure probabilities for switch and edge server, the average availability level of the task allocation scheme is not easily guaranteed. However, although the average unavailability level of each approach increases as the failure range increases (i.e., from range1 to range2), the OTAA is still superior to other approaches.

3) Effect of the Number of Base Stations

As shown in Fig. 8, these figures display the impact of number of base stations on the unavailability level, resource wastage, and communication delay. To show this effect more clearly, the number of edge servers varied accordingly, i.e., 76, 101, 127, and 152; the number of the virtual machines was set at 60; the number of the IoT applications was set at 20; the tunable positive weights $\theta_1,\theta_2,\theta_3$ were all set at 1/3; the failure probabilities of the edge server and switch were randomly selected from the range1 (i.e., [0.02, 0.12] and [0.05, 0.15]), respectively; and the number of base stations increased from 15 to 30 according to 5. By analyzing these figures, we can observe that the average resource wastage fluctuates as the number of base stations increases, and can be thought of as not being much affected. This is due to that the virtual machines need to be reallocated as the number of base stations changes. Meanwhile, the average unavailability level decreased as the number of base stations increases, and the average communication delay hardly changed as the number of base stations increases. This is due to that there are more edge servers with a low failure probability as the number of base stations increases. The average unavailability level, average resource wastage, and communication delay using the OTAA are both the lowest of all approaches.

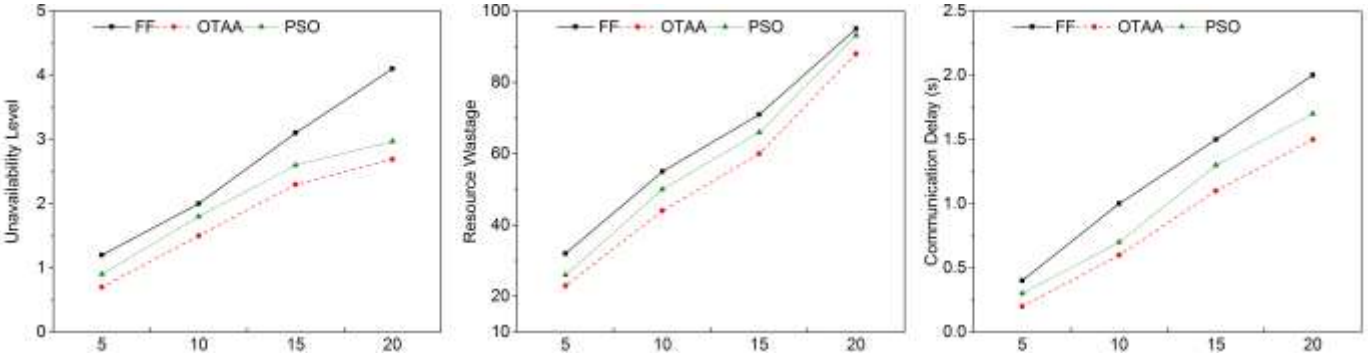
4) Effect of the Number of IoT Applications

As shown in Fig. 9, these figures display the impact of number of IoT applications on the unavailability level, resource wastage, and communication delay. To show this effect more clearly, the number of base stations and edge servers was set at 25 and 127, respectively; the failure probabilities of the edge server and switch were randomly selected from the range1 (i.e., [0.02, 0.12] and [0.05, 0.15]), respectively; the tunable positive



(a) Impact on the unavailability level (b) Impact on the resource wastage (c) Impact on the communication delay

Fig. 8. The effect of different number of base stations. The number of base stations represents how many base stations can be in the mobile edge computing environment. Although the average resource wastage fluctuates as the number of base stations increases, it can be thought of as not being much affected; the average unavailability level decreased as in the number of base stations increases; the average communication delay fluctuates as the number of base stations increases.



(a) Impact on the unavailability level (b) Impact on the resource wastage (c) Impact on the communication delay

Fig. 9. The effect of different number of IoT applications. The number of IoT applications represents how many IoT applications can be offloaded to the edge clouds. The average unavailability level, average resource wastage, and average communication delay increased as the number of IoT applications increases.

weights $\theta_1,\theta_2,\theta_3$ were all set at 1/3; the number of IoT applications increased from 5 to 20 according to 5; the number of virtual machines increased from 15 to 60 according to 15. By analyzing these figures, we can observe that the average unavailability level, average resource wastage, and average communication delay increased as the number of IoT applications increases. The average unavailability level, average resource wastage, and average communication delay using the OTAA are both the lowest of all approaches.

VII. Conclusions and Future Work

With the increasing popularity of the mobile edge computing, Edge clouds have become common platforms for offloading multiple collaborative tasks of the IoT application. Enhancing the availability level and resource utilization of the allocation scheme of these tasks under the condition of certain communication delay has become a matter of great concern. In this paper, we proposed and mathematically established three models: 1) one formulates the unavailability level by considering the edge severs and switches with heterogeneous failure probabilities; 2) another formulates the communication delay while dealing with the IoT applications; 3) the third formulates the resource wastage. We also proposed a joint optimization objective to simultaneously minimize the unavailability level and resource wastage under the condition of certain communication delay by our proposed approach based on BBO algorithm. Finally, we carried out the simulation experiments to demonstrate the performance and effectiveness of our proposed approach.

In our future work, we will extend the above optimization problem via offloading the tasks of IoT application to the remote cloud, and then investigate the influence of the sequence of task allocation and the dynamic change of these tasks after allocation on the above optimization objectives.

Acknowledgment

This work is supported by the National Key R&D Program of China (2018YFE0205503), NSFC (61922017, 62032003, and 61921003), the Key Science and Technology Research Project of Henan Province (192102310212, 202102210163, and 202102210152), the Key Science and Technology Research Project of Anyang City (2021C01GX017), and the Research and Cultivation Fund Project of Anyang Normal University (AYNUKPY-2019-24). Shangguang Wang is the corresponding author.

References

[1] J. Ni, K. Zhang, X. Lin, and X. S. Shen, "Securing Fog Computing for Internet of Things Applications: Challenges and Solutions," IEEE Communications Surveys and Tutorials, 2018, 20 (1): 601-628.

[2] Cisco V. Cisco Visual Networking Index: Forecast and Trends, 2017–2022. White Paper, 2018.

[3] X. Sun and N. Ansari, "EdgeIoT: Mobile Edge Computing for the Internet of Things," IEEE Communications Magazine, 2016, 54(12): 22-29.

[4] E. Ahmed, A. Ahmed, I. Yaqoob, J. Shuja, A. Gani, M. Imran, and M. Shoaib, "Bringing Computation Closer

toward the User Network: Is Edge Computing the Solution?" IEEE Communications Magazine,2017, 55(11): 138-144.

[5] S. Wang, R. Urgaonkar, M. Zafer, T. He, K. Chan, and K. K. Leung, "Dynamic Service Migration in Mobile Edge-Clouds," IEEE 2015 IFIP Networking Conference (IFIP Networking 2015), 2015,online publishing.

[6] A. Aral and I. Brandic, "Quality of Service Channelling for Latency Sensitive Edge Applications," Proceedings of IEEE International Conference on Edge Computing (EDGE 2017), 2017, pp. 166-173.

[7] M. Soualhia, C. Fu, and F. Khomh, "Infrastructure fault detection and prediction in edge cloud environments," Proceedings of the 4th ACM/IEEE Symposium on Edge Computing (SEC 2019), 2019, pp. 222-235.

[8] C. Hong and B. Varghese, "Resource Management in FogEdge Computing A Survey on Architectures, Infrastructure, and Algorithms, "ACM Computing Serveys, 2019, 52(5):97:1-97:37.

[9] T. Tran, A. Hajisami, P. Pandey, and D. Pompili, "Collaborative Mobile Edge Computing in 5G Networks: New Paradigms, Scenarios, and Challenges," IEEE Communications Magazine, 2017, 55(4): 54-61.

[10] J. Shuja, A. Gani,K. Ko,K. So, S. Mustafa, S.Madani, and M. Khan, "SIMDOM: A framework for SIMD instruction translation and offloading in heterogeneous mobile architectures," Transactions on Emerging Telecommunications Technologies, 2018, 29(4):e3174.

[11] X. Liu, S. Sun, and G. Huang. "Decentralized Services Computing Paradigm for Blockchain-Based Data Governance: Programmability, Interoperability, and Intelligence," IEEE Transactions on Services Computing, 2020, online publishing.

[12] H. Zhu and C. Huang, "Availability-Aware Mobile Edge Application Placement in 5G Networks," IEEE Global Communications Conference (Globecom 2017), 2017, online publishing.

[13] D. Simon, "Biogeography-based optimization," IEEE Transaction on Evolutionary Computation, 2008, 12(6): 702-713.

[14] J. Yao and N. Ansari, "Fog Resource Provisioning in Reliability-Aware IoT Networks," IEEE Internet of Things Journal, 2019, 6(5): 8262-8269.

[15] A. M. Maia, Y. Ghamri-Doudane, D. Vieira, and M. F. de Castro, "A Multi-Objective Service Placement and Load Distribution in Edge Computing," IEEE Global Communications Conference (Globecom 2019), 2019, online publishing.

[16] J. Oueis, E. C. Strinati, and S. Barbarossa, "The Fog Balancing: Load Distribution for Small Cell Cloud Computing," IEEE Vehicular Technology Conference (VTCSpring 2015), 2015, online publishing.

[17] L. Zhao and J. Liu, "Optimal Placement of Virtual Machines for Supporting Multiple Applications in Mobile Edge Networks," IEEE Transactions on Vehicular Technology, 2018, 67 (7):6533-6545.

[18] B. Hu, J. Chen, and F. Li, "A dynamic service allocation algorithm in mobile edge computing," International Conference on Information and Communication Technology Convergence (ICTC 2017), 2017, online publishing.

[19] J. Xie, C. Qian, D. Guo, X. Li, S. Shi, and H. Chen, "Efficient Data Placement and Retrieval Services in Edge Computing," IEEE 39th International Conference on Distributed Computing Systems (ICDCS 2019), pp. 1029-1039, 2019.

[20] S. Pasteris, S. Wang, M. Herbster, and T. He, "Service placement with provable guarantees in heterogeneous edge computing systems," IEEE Conference on Computer Communications (INFOCOM 2019), pp. 514-522, 2019.

[21] V. Farhadi, F. Mehmeti, and T. Porta, "Service Placement and Request Scheduling for Data-intensive Applications in Edge Clouds," IEEE Conference on Computer Communications (INFOCOM 2019), pp. 1279-1287, 2019.

[22] Y. Chen, S. Deng, H. Zhao, Q. He, Y. Li, and H. Gao, "Data-intensive application deployment at edge: A deep reinforcement learning approach," IEEE International Conference on Web Services (ICWS 2019), pp. 355-359, 2019.

[23] J. Meng, H. Tan, C. Xu, W. Cao, L. Liu, and B. Li, "Dedas: Online task dispatching and scheduling with bandwidth constraint in edge computing," IEEE Conference on Computer Communications (INFOCOM 2019), pp. 2287-2295, 2019.

[24] Y. Chen, S. Deng, H. Ma, and J. Yin, "Deploying Data-intensive Applications with Multiple Services Components on Edge," Mobile Networks and Applications, 2020, 25(2): 426-441.

[25] A. Khan, M. Othman, A. Khan, J. Shuja, and S. Mustafa, "Computation Offloading Cost Estimation in Mobile Cloud Application Models," Wireless Personal Communications, 2017, 97(3):4897-4920.

[26] X. Guo, R. Singh, T. Zhao, and Z. Niu, "An index based task assignment policy for achieving optimal power-delay tradeoff in edge cloud systems," IEEE International Conference on Communications (ICC 2016), 2016, online publishing.

[27] A. M. Maia, Y. Ghamri-Doudane, D. Vieira, and M. F. de Castro, "Optimized Placement of Scalable IoT Services in Edge Computing," IFIP/IEEE Symposium on Integrated Network and Service Management (IM 2019), pp. 189-197, 2019.

[28] M. Goudarzi, H. Wu, M. S. Palaniswami, and R. Buyya, "An Application Placement Technique for Concurrent IoT Applications in Edge and Fog Computing Environments," IEEE Transactions on Mobile Computing, 2020, 20 (4): 1298-1311.

[29] K. Peng, H. Huang, S. Wan, and V. Leung, "End-edge-cloud collaborative computation offloading for multiple mobile users in heterogeneous edge-server environment," Wireless Network, 2020, online publishing.

[30] S. Cheng, Z. Chen, J. Li, and H. Gao, "Task Assignment Algorithms in Data Shared Mobile Edge Computing Systems," IEEE 39th International Conference on Distributed Computing Systems (ICDCS 2019), pp. 997-1006, 2019.

[31] J. Oueis, E. Calvanese-Strinati, A. De Domenico, and S. Barbarossa, "On the impact of backhaul network on

distributed cloud computing," IEEE Wireless Communications and Networking Conference Workshops (WCNCW 2014), pp. 12-17, 2014.

[32] G. Huang, C. Luo, K. Wu, Y. Ma, Y. Zhang, and X. Liu . "Software-Defined Infrastructure for Decentralized Data Lifecycle Governance: Principled Design and Open Challenges," Proceeding ofIEEE 39th International Conference on Distributed Computing Systems (ICDCS 2019), 2019, online publishing.

[33] P. Gill, N. Jain, and N. Nagappan, "Understanding network failures in data centers: measurement, analysis, and implications," Proceeding of ACM Conference on Special Interest Group on Data Communication (SIGCOMM 2011), pp. 350-361, 2011.

[34] 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," Journal of Computer and System Sciences, 2013, 79(8): 1230-1242.

[35] J. Xu and J. A. Fortes, "Multi-objective virtual machine placement in virtualized data center environments," Proceeding of IEEE/ACM International Conference on Green Computing and Communications (GreenCom 2010), pp. 179-188, 2010.

[36] S. Srikantaiah, A. Kansal, and F. Zhao, "Energy aware consolidation for cloud computing," Cluster Computing, 2008, 12(1):10-10.

[37] H. Ma, D. Simon, P. Siarry, Z. Yang, and M. Fei, "Biogeography-Based Optimization: A 10-Year Review," IEEE Transactions on Emerging Topics in Computational Intelligence, 2017, 1(5): 391-407.

[38] J. Liu, S. Wang, A. Zhou, R. Buyya, and F. Yang, "Availability-aware Virtual Cluster Allocation in Bandwidth-constrained Datacenters," IEEE Transactions on Services Computing, 2020, 13(3): 425-436.

[39] H. Ma, S. Ni, and M. Sun, "Equilibrium species counts and migration model tradeoffs for biogeography-based optimization, Proceeding of IEEE 48th International Conference on Decision and Control (CDC 2009), pp. 3306-3310, 2009.

[40] J. Liu, S. Wang, A. Zhou, S. Kumar, F. Yang, and R. Buyya, "Using Proactive Fault-tolerance Approach to Enhance Cloud Service Reliability," IEEE Transactions on Cloud Computing, 2018, 6(4):1191-1202.

[41] H. Gupta, A. Vahid Dastjerdi, S. K. Ghosh, and R. Buyya, "ifogsim: A toolkit for modeling and simulation of resource management techniques in the internet of things, edge and fog computing environments," Software: Practice and Experience, 2017, 47(9): 1275–1296.

[42] 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 and Computation: Practice and Experience, 2012, 24(13): 1397-1420.

[43] D. Simon, M. Ergezer, and D. Du, "Population distributions in biogeography-based optimization algorithms with elitism," Proceeding of IEEE International Conference on Systems, Man and Cybernetics (SMC 2009), pp. 991-996, 2009.

[44] S. Wang, Z. Liu, Z. Zheng, Q. Sun, and F. Yang, "Particle swarm optimization for energy-aware virtual machine placement optimization in virtualized data centers," Proceeding of IEEE International Conference on Parallel and Distributed Systems (ICPADS 2013), pp. 102-109, 2013.

Jialei Liu is an assistant professor at School of Software Engineering, Anyang Normal University, China. He received the PhD degree in computer science and technology from Beijing University of Posts and Telecommunications (BUPT) in 2018. He received his ME in computer science and technology from Henan Polytechnic University. His major research interests include cloud computing and mobile edge computing.

Chunhong Liu is an associate professor in Department of Computer and Information Engineering at Henan Normal University, China. She received the PhD degree in computer science and technology from Beijing University of Posts and Telecommunications (BUPT) in 2018. She received her ME in computer science and technology from Xidian University. Her major research interests include cloud computing, edge computing, machine learning and oriented-service computing.

Bo Wang is a lecturer at School of Software Engineering, Anyang Normal University, China. He received the PhD degree in computer science and technology at Beijing Institute of Technology (BIT) in 2019. He received his ME degree in computer software and theory from Zhengzhou University. His major research interests include big data processing technology, cloud computing, computer system architecture and data mining.

Guowei Gao is an associate professor at School of Software Engineering, Anyang Normal University, Anyang, China. He received the PhD degree in information and communication engineering with Hohai University in 2018. He received the B.S. degree in computational science and the M.S. degree in applied mathematics from Henan University in 2005 and 2008, respectively. His major research interests include signal processing, edge computing, data mining and machine learning.

Shangguang Wang received his Ph.D. degree at Beijing University of Posts and Telecommunications in 2011. He is currently a professor and deputy director at the State Key Laboratory of Networking and Switching Technology, BUPT. He has published more than 100 papers, and played a key role at many international conferences, such as general chair and PC chair. His research interests include edge computing, service computing, and cloud computing. He is a senior member of the IEEE, and the Editor-in-Chief of the International Journal of Web Science.