IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 28, 2020, accepted April 16, 2020, date of publication April 21, 2020, date of current version May 11, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2989154

QoS Guaranteed Resource Allocation for Live Virtual Machine Migration in Edge Clouds

LEI YANG[1], (Member, IEEE), DOUDOU YANG[1], JIANNONG CAO[2], (Fellow, IEEE), YUVRAJ SAHNI[2], AND XIAOHUA XU[3], (Member, IEEE)

[1]School of Software Engineering, South China University of Technology, Guangzhou 510641, China
[2]Department of Computing, The Hong Kong Polytechnic University, Hong Kong
[3]Department of Computer Science, Kennesaw State University, Kennesaw, GA 30144, USA

Corresponding author: Lei Yang (sely@scut.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61972161, in part by the Hong Kong RGC General Research Fund under Grant PolyU 15217919 and Grant PolyU 152133/18, in part by the Key Research and Development Program of Guangdong Province under Grant 2019B010154004, in part by the Guangdong Basic and Applied Basic Research Foundation under Grant 2020A1515011496, and in part by the Fundamental Research Funds for the Central Universities under Grant 2018MS53.

ABSTRACT Live Virtual Machine (VM) migration among geographically distributed edge clouds is an important strategy for providing low latency and reliable services for mobile end users. VM migration among edge clouds is more challenging than that in cloud computing, because the network bandwidth among edge clouds is more constrained than the cloud data center networks. In this paper, we study the bandwidth allocation among multiple concurrent live VM migrations in edge clouds. This problem is novel in that existing works aim to reduce the migration time for a single live VM migration among the edge clouds, and also ignores the QoS requirement for the service running on the VM in migration. However, our problem considers multiple VM migration tasks, and aims to maximize the average QoS while meeting the migration time constraint for each VM migration task. We formulate the problem as a Non-Linear Programming (NLP) problem which is also shown to be NP-Hard. We develop a new method to solve this problem. In our approach, we first transfer the problem into a Linear Programming (LP) problem by reducing the solution space. Taking the output from the LP solver as an initial solution, we then develop a heuristic to adjust it in order to find a better one to the original NLP problem. Finally, we design a set of evolutionary algorithms to select the optimal initial solution from the LP solver. Extensive simulations show that our proposed method can achieve good QoS and also has a fast convergence speed.

INDEX TERMS Edge cloud, live VM migration, QoS, resource allocation.

I. INTRODUCTION

Edge cloud is a pool of virtualized and configurable computing resources at the network edge which are physically closer to the end users than the Internet cloud. The edge clouds are lightweight and usually distributed geographically across various locations [1], [2]. With the advantage of access latency, edge clouds will become important infrastructures in future for many vertical applications including industry IoT, automated driving, intelligent transportation and so on [3], [4]. It is reported that industrial enterprises including Amazon, Microsoft, Google, Cisco and Huawei have planned their products on the market of edge clouds.

The academic research on edge clouds has increased dramatically in recent years. One fundamental problem is service migration among the geo-distributed edge clouds [5]. The purpose of migration is to provide low service latency as the end user moves across different places. Another purpose is to guarantee the reliability in case some of the edge clouds have failures. To provide un-interrupted services, the live service migration is needed across the edge clouds, which requires that the service is able to keep running normally during the migration. There exist several implementation techniques to support service migrations such as Virtual Machine (VM), container and uni-kernel. Although container [6] and uni-kernel [7] are more lightweight, they can not support the live migration well as VM does.

Several mature approaches such as pre-copy [8] and post-copy [9], [10] to support the live VM migration have been

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

proposed for cloud data centers. Among these approaches, pre-copy is more widely used in the cloud VM migration. Using this approach, the migration time is approximately to be the state size divided by the network bandwidth minus the dirty (service) rate [11], where dirty rate is a data rate that the state in the memory of a VM is updated in migration. The dirty rate depends on the QoS requirement of the service running in VM. A high QoS requirement causes a high dirty rate accordingly and thus needs much bandwidth resource. Live VM migration with a low migration time and high QoS among the edge clouds is more challenging than that in traditional cloud data centers. This is because the network connecting the edge clouds, i.e., cellular network, WLAN or wide area network, has a more constrained bandwidth than data center networks [12].

In this paper, we study the resource allocation problem for multiple live VM migrations across the edge clouds. The problem is to allocate the bandwidth and determine the service rate for the VM migration tasks such that the average QoS of the migration tasks is maximized and the migration time meets the deadline. The problem is novel in two aspects. First, most research mainly focus on the performance optimization of a single VM migration [12], [13], while our problem takes into account the concurrent multiple VM migrations caused by the user mobility in edge clouds. Second, previous research aims to reduce the migration time and ignores the QoS which is a significant metric concerned by the user in live migration [12]–[14]. The metrics, i.e., migration time and QoS, are two conflicting objectives, and thus make our problem more difficult.

We formulate the resource allocation problem into a Non-Linear Programming Problem (NLP). To solve the problem, we first transform the problem into a Linear Programming (LP) problem by assuming that all the migration tasks are completed exactly at the deadlines. We then design a new model of resource transaction, and use the model to iteratively adjust the solution from the LP solver. The adjustment starts with an initial solution from LP solver, and aims to find an optimal solution to the original problem. The resource transaction model proposed by us can guarantee that the solution after an adjustment is still feasible. Since the output of the LP solver affects the final results, we finally propose a set of evolutionary algorithms to select the optimal one from the solutions generated by the LP solver. We evaluate the performance of our approach via extensive simulations. In particular, we compare the performance of five evolutionary algorithms used to select the initial solution in our approach. The results show that our proposed approach can achieve good QoS and has a fast convergence speed under various simulation settings. The contributions of this paper are as follows.

- To the best of our knowledge, this work is the first one to study the resource allocation problem for multiple live VM migration in edge clouds, where live VM migration means that the VM still provide services for users during migration. The problem is novel in the trade-off between QoS and migration time for multiple concurrent VM migration tasks.
- We develop a new approach to solve this problem. The novelty of this approach is to design a resource transaction model to iteratively search the optimum from an initial solution.
- We have done extensive simulations, and validated that our approach has good performance in average QoS and convergence time.

II. SYSTEM MODEL AND PROBLEM FORMULATION

A. SYSTEM MODEL

Fig.1 shows the system model. We consider a 2-D area deployed with a set of edge clouds distributed in different locations. Each edge cloud has its own coverage area. The role of each edge cloud is to provide the host environment for the virtual machines. The virtual machines provide a type of services for the mobile end users, such as video streaming, mobile gaming, message transfer service and so on. Service is an abstract concept, which can be referred to any application executed on the VM. In this system, the end users request the services running on the VMs at different edge clouds. End users always move around, and sometimes they move out of the coverage area of the edge cloud hosting their requested services. The roaming of users would lead to multiple VM migrations among the edge clouds. Meanwhile, the user requires that the service can still run with high QoS while the migrations happen.

For example, in mobile gaming, the service of 3D rendering is deployed on the VM in the edge cloud for each end user. The resolution of a rendering picture represents the quality of this service. In a video streaming application, the VM in the edge cloud hosts a broker service to transform the content formats for the end user. The resolution of the content delivered to the end user represents the quality of the service. If the QoS requirement is higher, the migration of the VM hosing the service would need more network bandwidth. Besides being caused by the user mobility, VM migration also occurs for load balancing among the edge clouds and fault tolerance. To avoid the service disruption during the VM migration, we assume that a pre-copy technique [8] has been implemented for live migration.

We consider a live VM migration from a source machine in an edge cloud to a destination machine in another edge cloud. During the migration, we assume that the state in the memory of the VM needs to be transmitted to the destination machine. Since the service on the VM can still run during the migration, the memory of the VM on the source machine would be updated when the state is transmitted to the destination. Thus, the live migration technique like pre-copy iteratively transmits the content in the memory from the source to the destination machine. We concern about two objective metrics of a live VM migration: migration time and Quality of Service (QoS). Migration time indicates how much time it takes for

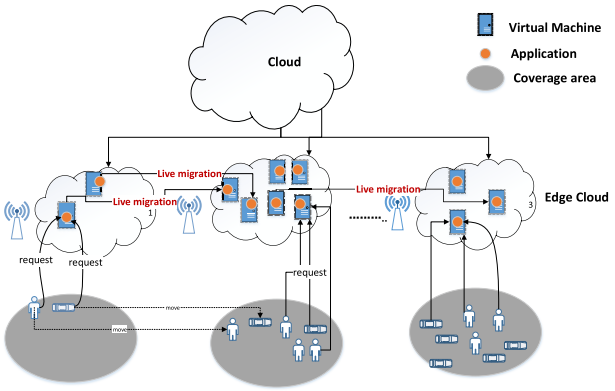


FIGURE 1. The system model.

a VM to migrate from an edge cloud to another. QoS is an indicator of the quality requirement for the service running on a VM in migration. If a service user demands a high QoS, the memory of the VM will be updated with a high data rate during the migration. We name this rate as memory *dirty rate*. Since dirty rate determines the quality of service, we use *dirty rate* and *service rate* interchangeably. By using the pre-copy technique, the VM migration time can be modeled by

$$MT = \frac{M}{B - d} \tag{1}$$

where MT is the migration time of the live VM migration, M is the size of the VM memory, B is the network bandwidth between the source and destination machines, and d is the dirty rate [11].

In mobile edge computing, the edge clouds are normally deployed in a cellular backhaul network. The backhaul bandwidth among the edge clouds directly affects the VM migration time and QoS. A VM migration demanding low migration time and high QoS should be allocated with sufficient bandwidth. In real environments, there exist a large number of concurrent VM migration tasks, the problem is how to allocate the constrained bandwidth to each VM migration task such that the average QoS is maximized and the migration time constraint is satisfied.

B. PROBLEM FORMULATION

In the problem, we assume that the time period is calculated by time slots, and the total number of time slots is T. We use j to index the time slot. The number of the VM migration tasks during this time period is N. We use i to index the task. Each VM migration task i has its own amount of data to be transmitted, denoted by D_i. Normally D_i is equal to the memory size of the VM. Every VM migration task has its own predefined deadline denoted by f_i. All migration tasks compete for the network bandwidth across the edge clouds. In reality, the bandwidth changes with time, and we use B_j to denote total bandwidth across the edge clouds at time j. We use $n_{i,j}$ to denote the bandwidth allocated for migration task i at time slot j. Meanwhile, the VM still provides service to users during migration. We use $d_{i,j}$ to denote service rate of the migration task i at time slot j. The QoS is determined by this service rate.

TABLE 1. Mathematical notations in this paper.

i	index of live virtual machine migration task;
j	index of each time slot;
t	length of each time slot;
T	total number of time slots;
N	the number of live virtual machine migration tasks in T;
B_j	network bandwidth provided for all virtual machine migration tasks at time slot j;
D_i	the workload of data transmission of the VM migration task i, which is usually equal to the memory size of the VM;
$Q_{i,j}$	the QoS of the VM migration task i at the time slot j;
m	minimum QoS requirement for the VM migration task;
$n_{i,j}$	allocated bandwidth for migration task i at time slot j;
$d_{i,j}$	service rate (or dirty rate) of migration task i at time slot j;
f_i	deadline of the migration time for the VM migration task i;
l_i	actual migration time of the VM migration task i;

We assume the relationship between QoS and service rate is positively linear, which is shown in Equation (3). The QoS is the application level quality of the service running on the VM in the edge cloud. When a VM migrates from an edge cloud to another, the services running on the VM would cause a dirty rate of its memory. A high QoS would lead to a larger dirty rate on the memory. In the applications like mobile gaming and 3D rendering, the QoS is measured by the data size per second, while the dirty rate is also measured by data rate, i.e., MB/s. We can observe that the dirty rate increases linearly with the required QoS in these applications. In the other applications, the QoS would increase positively with the dirty rate, but the relationship may not be linear. However, for simplicity, we use a linear model in the problem formulation. Since our proposed algorithms do not require the linear model of the QoS and dirty rate, our proposed algorithms still work under the other models. So this assumption does not affect the general usability of our proposed algorithms.

We now formulate the problem in Equation (2).The problem has three types of decision variables. The first one is a continuous variable $n_{i,j}$ that represents bandwidth allocation. The second one is another continuous variable $d_{i,j}$ that represents the service rate of migration task i at time slot j. The third one is a discrete variable l_i, which represents the actual completion time of migration task i.

Next, we model the constraints of this problem as follows. First, we need to ensure that for each virtual machine migration task, the QoS should satisfy the minimum requirement at each time slot j. This constraint is formulated by Equation (4). Second, for each time slot j, the VM migration tasks compete for the total bandwidth resource B_j. So the sum of allocated bandwidth for the migration tasks can not exceed the total bandwidth B_j. This constraint is formulated by Equation (5). For every migration task, the sign of completion is that all of its amount of data has been transmitted. This constraint is

formulated by Equation (6).Finally, for every VM migration task, the actual completion time should not exceed a predefined deadline. This constraint is formulated by Equation (7). We formulate the resource allocation problem by

$$\max_{n_{i,j},d_{i,j},l_i} \frac{1}{N}\sum_{i=1}^{N}(\frac{1}{l_i}\sum_{j=1}^{l_i} Q_{i,j}) \tag{2}$$

$$\text{s.t. } Q_{i,j} = f(d_{i,j}) \quad i = 1, 2, \ldots, N; j = 1, 2, \ldots, l_i \tag{3}$$

$$\forall i \in [1, N], \quad \forall j \in [1, l_i], \ Q_{i,j} \geq m \tag{4}$$

$$\forall j \in [1, l_i], \ \sum_{i=1}^{N} n_{i,j} \leq B_j \quad i = 1, 2, \ldots, N \tag{5}$$

$$\forall i \in [1, N], \ \sum_{j=1}^{l_i}(n_{i,j} - d_{i,j}) * t = D_i \tag{6}$$

$$\forall i \in [1, N], \quad l_i \leq f_i \tag{7}$$

The objective of this problem is to make decisions for these three variables, i.e., $n_{i,j}$, $d_{i,j}$ and l_i, such that the average QoS of the VM migration tasks is maximized.

III. APPROACH TO THE PROBLEM

Our problem has three decision variables: $n_{i,j}$, $d_{i,j}$ and l_i. The problem is a non-linear programming problem, since the decision variable l_i appears as a denominator in Equation (2). It is very difficult to solve such problem by using existing optimization solvers. So we develop a novel and effective method to solve the problem. The overall idea of our method is as follows.

As shown in Equation (7), the actual migration time l_i should be less than a deadline. Suppose that every migration task is finished exactly at the required deadline, i.e., $l_i = f_i$, the problem is then transformed into a Linear Programming (LP) problem. Taking a feasible solution of the LP problem as an initial solution, we then develop an efficient heuristic (Algorithm 1) to adjust the solution $n_{i,j}$, $d_{i,j}$ with the aim of maximizing the increase of average QoS. During the adjustment, it is allowed that the migration time l_i can be advanced before the deadlines. As the initial solution from the LP problem significantly affects the results of our problem, finally we develop several evolutionary algorithms including the genetic algorithm and particle swarm optimization to find a good initial solution.

A. TRANSFORM INTO LINEAR PROGRAMMING (LP) PROBLEM

We first transform the original problem into a linear programming problem by assigning the variable l_i with the f_i. The transformed problem is as follow.

$$\max \frac{1}{N}\sum_{i=1}^{N}(\frac{1}{f_i}\sum_{j=1}^{f_i} Q_{i,j}) \tag{8}$$

$$\text{s.t. } Q_{i,j} = f(d_{i,j}) \quad i = 1, 2, \ldots, N; j = 1, 2, \ldots, f_i \tag{9}$$

$$\forall i \in [1, N], \forall j \in [1, f_i], \quad Q_{i,j} \geq m \tag{10}$$

$$\forall j \in [1, f_i], \ \sum_{i=1}^{N} n_{i,j} \leq B_j \quad i = 1, 2, \ldots, N \tag{11}$$

$$\forall i \in [1, N], \quad \sum_{j=1}^{f_i}(n_{i,j} - d_{i,j}) * t = D_i \tag{12}$$

The LP problem above includes two decision variables $n_{i,j}$ and $d_{i,j}$. Obviously the optimal solution of LP exists on the boundary of the solution space. In this LP problem, we can easily prove that the feasible solution locating at the boundary of solution space have the same values on the objective function. Thus, an arbitrary $n_{i,j}$ and $d_{i,j}$ which satisfies the Equations (12)(13) would be an optimal solution from the LP solver. This solution will be taken as an initial solution for subsequent adjustment.

$$\forall j \in [1, f_i], \quad \sum_{i=1}^{N} n_{i,j} = B_j \quad i = 1, 2, \ldots, N \tag{13}$$

B. HEURISTIC FOR ADJUSTING THE INITIAL SOLUTION FROM LP

We develop a heuristic strategy to adjust the initial solutions from LP. The adjustment tries to advance the completion time of some migration tasks such that the average QoS in Equation (8) is increased as much as possible. The core mechanism of this adjustment is bandwidth exchange between the migration tasks.

Consider each of the migration task as a selfish person who has certain amount of bandwidth as his/her resources. The total amount of resources initially owned by the task i is defined by $\sum_{j=1}^{f_i} n_{i,j} \times t$, where $n_{i,j}$ denotes the bandwidth allocation in the initial solution. The migration tasks can use their resources to do transactions. A *transaction* is defined as two times of resource exchange between the same pair of tasks. At one time, if one task acquires the bandwidth resource from the other task, it should return the same amount of resources to the task at the other time. A transaction guarantees that the amount of resources during two times of exchange should be the same. For example, suppose task i obtains Δn bandwidth from i' at time j, then task i must return the resources to i' at the other time which could be at one time slot or multiple time slots. The amount of resources in this transaction is $\Delta n \times t$.

The purposes of the transaction model aims to ensure: 1) $n_{i,j}$ and $d_{i,j}$ is still a feasible solution of the original problem after a transaction occurs between two migration tasks; 2) the QoS of the tasks which participate in the transaction at least will not be decreased. The adjustment of the initial solution includes a series of transactions among the migration tasks. Furthermore, with this transaction model, we can prove an important theorem as follows.

Theorem 1: For the migration task, before we handle the task, the service rate $d_{i,j}$ and $n_{i,j}$ should satisfy the Equation (12). Suppose the task i acquires bandwidth resource from the

other task i' at a time j,where $j \leq f_i$. Because the amount of bandwidth resource is increased, the completion time of the migration task i would be ΔL ahead of the deadline f_i and we have

$$\sum_{j=1}^{f_i-\Delta L} [(n_{i,j} - d_{i,j}) \times t] + \Delta n \times t = D_i. \quad (14)$$

From Equation (12), we can obtain

$$\sum_{j=1}^{f_i-\Delta L} (n_{i,j} - d_{i,j}) \times t + \sum_{j=f_i-\Delta L}^{f_i} (n_{i,j} - d_{i,j}) \times t = D_i. \quad (15)$$

Replacing Equation (14) with the left side of Equation (12), we can have

$$\sum_{j=f_i-\Delta L}^{f_i} n_{i,j} \times t = \Delta n \times t + \sum_{j=f_i-\Delta L}^{f_i} d_{i,j} \times t. \quad (16)$$

The bandwidth resources during $[f_i - \Delta L, f_i]$ are released by the task i, which is indicated by the left hand of (15). The amount of bandwidth resources acquired from the other task at time j is $\Delta n * t$. Obviously, we can see that the left hand of (16) is greater than $\Delta n * t$. We conclude that the released resource of task i is greater than the resource acquired from the other task.

Therefore, task i first returns the same amount of resource back to the task i' at the time $[f_i - \Delta L, f_i]$. The remaining resources can be used to do one more transaction with another task. Through this transaction, the task i can increase the QoS at some time before the completion time $f_i - \Delta L$. Theorem 1 indicates the reason why advancing the completion time of a migration task could be able to improve the average QoS. Algorithm 1 presents a heuristic to do the transactions in order to increase the average QoS.

Algorithm 1 Heuristic Algorithm for Adjusting the Initial Solution

Input: An initial solution $(n^*_{i,j}, d^*_{i,j}, f_i)$ from the LP problem;
Output: $n_{i,j}$, $d_{i,j}$, l_i;
1: Let $\mathcal{T}$ be a set of the tasks, and $\mathcal{T}$ is initially assigned with all the migration tasks;
2: **while** Find the migration task i which has the minimum value of f_i **do**
3: **for** each time slot $j \in [1, f_i]$ **do**
4: Choose another task k from $\mathcal{T}$ which has the minimum data transmission workload at j;
5: Task i initiates a resource transaction with task k, i.e., 1) task i acquires $r \times n_{k,j}$ bandwidth from task k at time j; 2) task j returns back the same amount bandwidth resources to task k during the period $[f_i - L, f_i]$, where r is a constant percentage, and L is the decrease of migration time of task i due to this transaction;
6: Task i initiates a resource transaction with a randomly chosen task p, i.e, 1) task i gives the remainder resources during $[f_i - L, f_i]$ to task k; 2) task p returns back the amount of resources to i during the period from time j;
7: Update the completion time of task i: $f_i \leftarrow f_i - L$;
8: **end for**
9: Delete the migration task i from $\mathcal{T}$;
10: **end while**
11: **return** $n_{i,j}$, $d_{i,j}$, l_i;

C. EVOLUTIONARY ALGORITHMS FOR SELECTING THE INITIAL SOLUTION

We attempt to select the initial solution respectively by using the PSO and genetic algorithms. In the two algorithms, we have the same *encoding* method to represent a solution. A solution $(n_{i,j}, d_{i,j})$ is composed of a set of real numbers. We can treat a solution as a string, in which each character represents a real number. The string contains two parts, i.e., one denotes the bandwidth allocation $n_{i,j}$ and the other one denotes the service rate $d_{i,j}$. The two parts have the same length. In one part of the string, each task i has a segment of string to represent the allocated bandwidth or the service rate from time 1 to f_i. The length of the segment of string varies depending on the tasks, because the completion time f_i of the tasks are different among the tasks. With the encoding method, the length of a solution is $2 \times \sum_i^N \sum_{j=1}^{f_i}$.

1) PARTICLE SWARM OPTIMIZATION (PSO)

We use a hybrid PSO version proposed in existing work [21]. The method performs a mutation operation on the particle in each iteration. The mutation method greatly affects the performance and convergence of PSO algorithm, so we design three mutation methods and wish to obtain the optimal one via the simulation. It should be guaranteed that a solution after the mutation is still a feasible solution of the LP problem. The mutation methods are described as follows.

Mutation 1: In a solution, we randomly select two tasks and a time slot, and let the tasks exchange part of the bandwidth at the time slot. After the bandwidth exchange, the task which acquires bandwidth from the other one has an increase in the service rate accordingly such that the completion time does not change. The task which provides some of the bandwidth to the other one correspondingly has a decrease in the service rate.

Mutation 2 and Mutation 3: The methods of Mutation 2 and Mutation 3 have very similar idea. As shown in Fig.2, we select two tasks and firstly let the task do a bandwidth transaction at two time slots. Then, we update the service rate for the two tasks at the same time slots. The purpose of the updates is to guarantee that the completion time of the tasks is not affected. The difference between the Mutation 2 and Mutation 3 exists in how the tasks and time slots are chosen to do a transaction. In Mutation 2, we first choose the task which has the shortest completion time, and then randomly choose

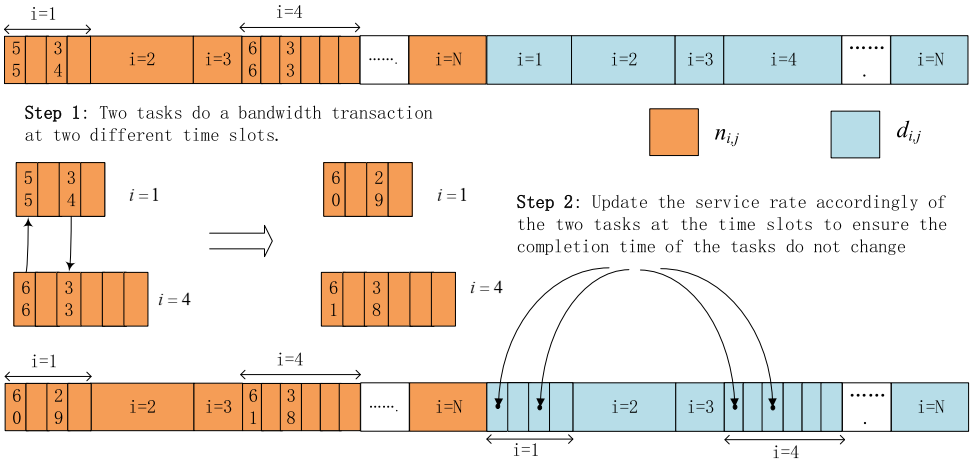


FIGURE 2. Illustration of the mutation.

two time slots. We calculate the difference of the allocated bandwidth between the two slots. The task which has the closest difference with the first chosen task will be selected as the other task. In Mutation 3, we first randomly select a time slot, and then calculate the difference of the bandwidth between any pair of tasks at the time slot. The pair of tasks which have maximum difference are chosen. According to the selected tasks, we then choose the other one time slot.

In summary, Mutation 1, Mutation 2, and Mutation 3 do bandwidth transaction between one pair of the migration tasks. The difference lies in the pattern of bandwidth transaction. Mutation 1 randomly selects two tasks and one time slot to do bandwidth transaction. Mutation 2 and Mutation 3 choose two tasks and two time slots to do bandwidth transaction. For the two chosen tasks, one of them receives certain amount of bandwidth from the other task at one time slot and returns the equal amount of bandwidth at the other time slot. In this way, we can make more changes to the solution compared to Mutation 1 while keeping the changed solution still feasible.

2) GENETIC ALGORITHM

In the genetic algorithm, we use the same mutation approach proposed for PSO. We design two crossover methods which are described as follows.

Crossover 1: Suppose two solutions A and B do a crossover. We first randomly select a migration task denoted by i, and two time slots j and j'. Then, we interchange the segment between A and B which represents the service rate of task i during period $[j, j']$ (Step 1 in Fig.3). After the interchange, we update respectively in A and B the corresponding bandwidth allocation of task i during the same time period (Step 2). The update ensures that the completion time of task i does not change. Due to the bandwidth update of task i, we update accordingly the bandwidth of all the other tasks expect i, so as to satisfy the bandwidth constraint at every time slot (Step 3). Finally, for each task with the bandwidth being updated, the service rate is updated such that the completion time maintains unchanged (Step 4).

Crossover 2: In the crossover method, we randomly select a migration task i and two time slots j and j'. Different with Crossover 1, we interchange both the segments between two chromosomal, e.g., A and B, which include the service rate and bandwidth allocation of task i during the period $[j, j']$. For each chromosomal, we either update the bandwidth or the service rate located at the interchanged segment, in order to guarantee that the completion time of task i does not change. Finally, we update the bandwidth of the other tasks to satisfy the bandwidth constraint.

IV. EVALUATION

We have done extensive simulations to compare the performance of the various proposed algorithms. We have five algorithms for comparison. The first three algorithms use the PSO to iteratively find the initial solution. They are different in the mutation mechanisms used. The algorithms are respectively denoted by PSO-1, PSO-2, and PSO-3. The other two algorithms apply the genetic algorithms to find the initial solution, while they have different crossover methods. These two algorithms are denoted by Gen-1 and Gen-2. The performance metrics for comparison include the average QoS and convergency time. Convergency time is measured by the number of iterations for the algorithm to converge. We compare the performance in five different parameter settings. Table 2 shows one of the parameter setting. In the table, the three parameters f_i, B_j and D_i are represented with the mean value and standard variance. Note that the deadline of each individual migration task is an integer measured by time units.

Table 3 shows the performance results of the algorithms under the five parameter settings. The last column presents the number of iterations the algorithms need to converge. For example, 28/55 means that the algorithm has 55 iterations in total, while it converges at the 28-th iteration. The table shows

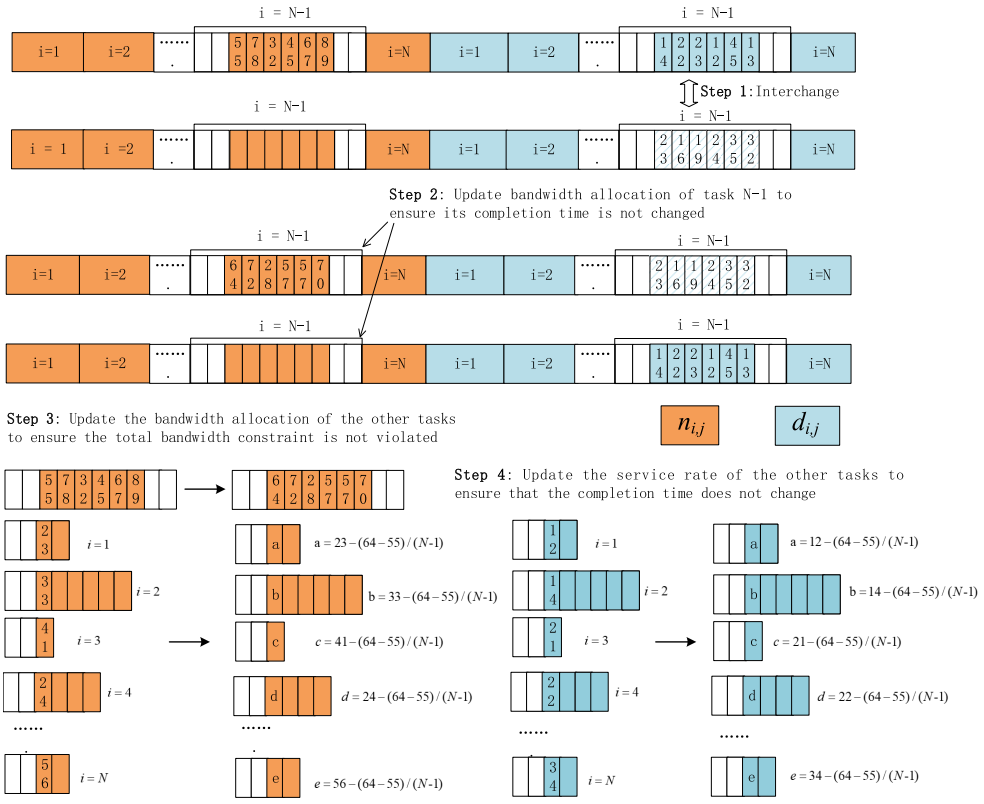


FIGURE 3. Illustration of the crossover in the genetic algorithm.

TABLE 2. Parameter setting in the simulation.

Parameters	Values
Length of a time slot t	2s
Total number of time slots T	60
Number of VM migration tasks N	78
Minimum constraint on the service rate at each time slot	10 MB/s
Deadlines f_i of the migration tasks	(41.6s, 11.3s)
Bandwidth B_j at a time slot	(2296, 477MB/s)
Data size of a migration task D_i	(1720, 1075MB)

that in various settings, the algorithms perform differently in both QoS and convergence time. Among the two genetic algorithms, Gen-1 has greater QoS than Gen-2 in most of the settings. The three PSO algorithms have very close performance in QoS. Moreover, all the algorithms can converge fast. Fig.4 and Fig.5 respectively show the average QoS and convergency time over all the five simulation settings. We can see that Gen-1 can achieve the greatest QoS among all the algorithms, while PSO-1 has the shortest convergency time.

Among all the five algorithms, Fig.4 shows that Gen-1 can achieve the greatest QoS, but we cannot objectively evaluate the algorithm itself. So we have added a simple baseline algorithm for comparison. In this baseline algorithm, we assume all VM migration tasks complete at their own deadlines. We first set the service rate of the migration task as the required minimum value. According to the amount of data for migration, for each migration task, we then increase the service rate at every time slot such that the migration is completed at the deadline. Under the five environment settings, the QoS obtained by the baseline algorithm are respectively 43.6203, 43.4412, 43.8259, 44.3839, and 43.6545. Fig.4 shows that our proposed algorithms have significantly higher QoS over the baseline algorithm.

Through the comparison with the simple and naïve baseline algorithm, we can show the efficiency of the proposed algorithms. As explained in our paper, the problem is a NP-Hard problem, searching the optimum in polynomial time is not feasible. The existing LP based solver could not efficiently solve the problem because the problem is non-convex. The proposed algorithms in this paper are heuristics. They have fast converge speed and can achieve much better solution than the baseline algorithm.

We evaluate how the bandwidth affects the performance of the algorithms. We change the bandwidth in a range from 1750MB/s to 1950MB/s, and keep the other parameters as the default values. Fig.6 shows the result of this simulation. We can see that the QoS increases as the edge cloud has an increasing bandwidth. This is because more bandwidth resources allow the migration task to increase the service rate while still maintaining the migration time unchanged.

TABLE 3. Simulation results.

	Algorithms	QoS	Migr. Time	Convergency
3*1	PSO-1	57.2369	40.6410	2/55
	PSO-2	58.1800	40.6538	28/55
	PSO-3	57.0436	40.7436	34/55
	Gen-1	59.9095	41.0897	3/55
	Gen-2	58.5505	40.9744	14/55
3*2	PSO 1	56.9630	39.1463	3/55
	PSO 2	56.9471	39.1829	49/55
	PSO 3	57.0524	39.1220	5/55
	Gen 1	56.6521	39.2805	31/55
	Gen 2	56.9077	39.5000	32/55
3*3	PSO 1	57.0546	41.0353	19/55
	PSO 2	56.7205	41.0824	1/55
	PSO 3	56.3828	41.2000	1/55
	Gen 1	57.4880	41.3176	5/55
	Gen 2	56.7148	41.3294	35/55
3*4	PSO 1	58.1166	39.2955	1/55
	PSO 2	58.1288	39.5682	24/55
	PSO 3	57.6758	39.3636	38/55
	Gen 1	59.0536	39.9773	25/55
	Gen 2	58.3600	39.9091	15/55
3*5	PSO 1	56.5081	40.9394	11/55
	PSO 2	57.5540	40.9242	27/55
	PSO 3	58.3661	40.9091	22/55
	Gen 1	57.8631	41.0152	18/55
	Gen 2	57.4315	41.0606	7/55

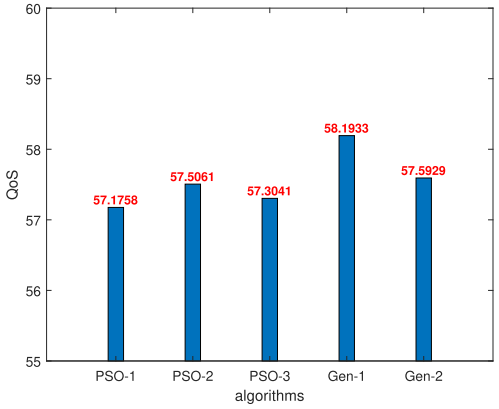


FIGURE 4. Overall performance results of the algorithms in QoS.

Among the five algorithms, PSO-3 has better performance than the other algorithms when the bandwidth increases to the maximum value. In the other cases, the five algorithms do not have much difference in the performance. Among the two genetic algorithms, we see that Gen-1 has slightly better performance then Gen-2 when the bandwidth is relatively great. This indicates that when doing the crossover in the genetic algorithm, the interchange of the service rate between two chromosomal works efficiently. It is not necessary to interchange the bandwidth allocation. This would simply the design of the genetic algorithm. For the three PSO algorithms, we can choose the simplest mutation method (PSO-1), which

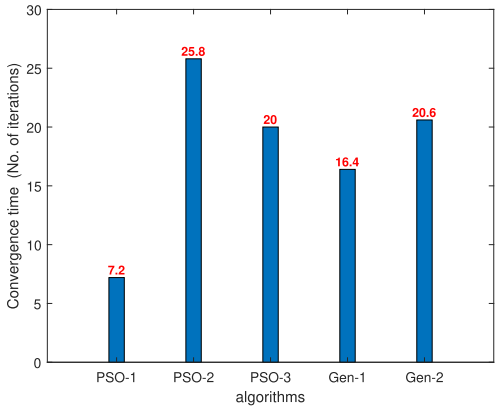


FIGURE 5. Overall performance results of the algorithms in convergence time.

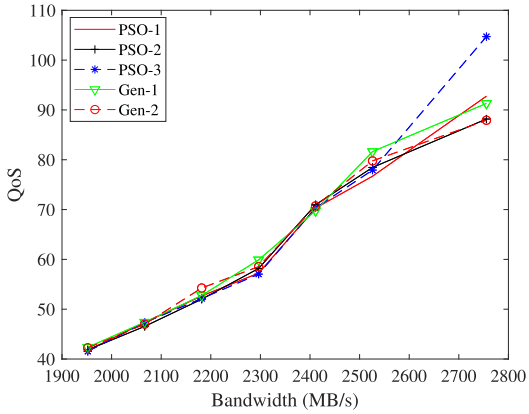


FIGURE 6. QoS changes depending on the bandwidth.

randomly changes the bandwidth between two migration tasks.

We then evaluate how the performance changes depending on the data volume of state D_i in migration. In the simulation, the average D_i ranges from 1375MB to 2235MB, and the other parameters are set as the default values. Fig.7 shows that the QoS decreases as the data volume increases. The data volume represents the load of data transmission during the VM migration. With the constrained bandwidth and the deadline on migration time, the increase of data volume must lead to the decrease of the service rate. In practical systems, the data volume of state is usually determined by the memory pages allocated for running the service in the VM. During the live VM migration, the memory pages need to be copied from the source server to the destination server. The QoS is determined by the dirty rate of the memory pages. This result implies that the service should be allocated with minimum memory pages as necessary in order to achieve a high QoS during live migration. Allocating unnecessary memory pages to a service should be avoided.

We evaluate how the performance changes depending on the deadlines of the migration time. In the simulation, we change the average value of f_i in the range [40.7s, 42.2s]. Fig.8 shows that the QoS does not have obvious pattern as the deadline increases. It implies that the changes of f_i within

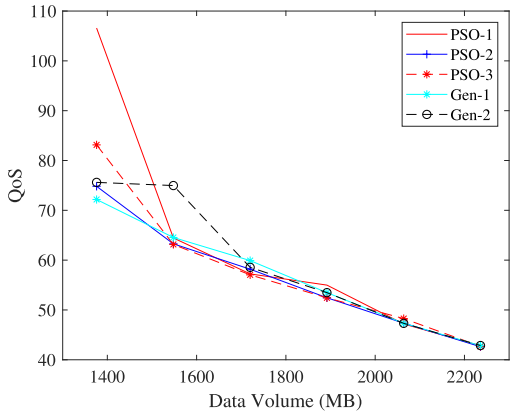


FIGURE 7. QoS changes depending on the data volume.

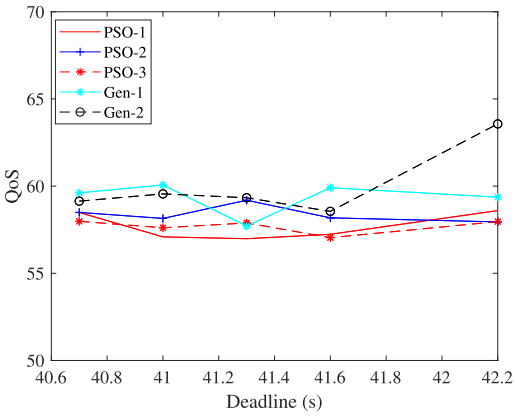


FIGURE 8. QoS changes depending on the deadlines.

this range do not affect the performance. From the theoretical analysis, if the deadline of every task increases, we can prove that the QoS must increase accordingly. The reason is that the optimal solution becomes one of the feasible solutions when the deadlines of the tasks increase. However, in this simulation, as the increase of the deadline represents the average value, some of the tasks have increasing deadlines, while some of the tasks have decreasing deadlines. It is not guaranteed that the QoS will increase finally.

V. RELATED WORKS

In this section, we present the existing research on service migration in edge clouds.

The service migration between edge clouds can be implemented by three techniques, which include Virtual Machine (VM), container [6], [15] and uni-kernel [7] based techniques. Although container and uni-kernel support more lightweight migration, they do not support well the live migration. Live migration requires that the service can not be interrupted while being in migration. However, there are several approaches to support live VM migration like pre-copy and post-copy methods [16]. Pre-copy method transfers the whole memory state in advance from the source server to the destination server. As some of the memory state may be updated during the transmission, the method may need to transfer the updated copy again. Therefore, the memory state of VM is transferred iteratively until the updated state is below a threshold. Post-copy method initiates the VM instance on the destination server first, and it pulls the state from the source server as it needs. In the pre-copy method, many skip techniques are used to optimize the migration time [9]. It predicts which part of the memory state is updated in the next iteration, and then skips the transmission of this part in the current iteration.

VM migration in edge clouds is much more difficult than that in the cloud, because the networks in edge cloud environment are mostly cellular networks or wide area network which have much more limited bandwidth than the cloud data center network [17]. Recently there have been many researches about live VM migration in edge clouds. Wang et al have a survey on the existing migration techniques in mobile edge computing [5]. Existing researches are mainly classified into three categories according to the objectives, i.e., migration time, service delay and service continuity. First, Machen et al [13] developed a layered framework for migrating active service applications using incremental file synchronization. In the framework, only those layers that are missing at the destination are transferred. Li et al [14] built a model to quantitatively predict live VM migration time, based on which, two optimal live VM migration strategies were then proposed to reduce the migration time. Chaufournier et al [12] proposed to use multi-path TCP to improve both the VM migration time and network transparency of applications.

Second, concerning about the server delay, Rodrigues et al [18] developed a strategy named by *Follow the Sun through the Clouds*. The strategy migrates key virtual machines of an application to a close place near the mobile user so as to reduce the service delay. Rodrigues et al [19] proposed an algorithm to optimize the resource utilization of cloudlets and the service delay by leveraging VM migration and transmission power control. Third, to guarantee the service continuity, Wang et al [5] proposed a strategy of service migration, which decided when or where the service was migrated based on the user mobility. Ha et al [20] proposed a framework and approach for adaptive VM migration between the cloudlets.

The existing researches above study how to improve the performance of a single VM migration task. In our work, we extend them to a scenario with multiple concurrent live VM migration tasks across different edge clouds. In such a scenario, the migration tasks compete for the constrained network bandwidth, so we focus on the bandwidth allocation among the VM migration tasks. Moreover, we consider how to guarantee the QoS of a live VM migration, while most of existing works on live VM migration only concern on the migration time.

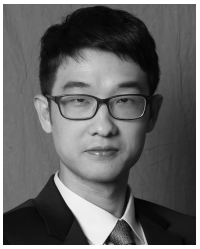
VI. CONCLUSION

In this paper, we have studied a novel problem of the resource allocation for multiple live VM migration in edge clouds.

The problem is to allocate the constrained bandwidth to the VM migration and meanwhile to determine a service rate at each time for the migration tasks. The objective is to maximize the average QoS during the migration and to meet the deadlines of migration time. We developed a new method to solve this challenging problem. In the method, the original problem is firstly transferred into a LP problem. We designed a resource transaction based heuristic to adjust a initial solution from the LP. We developed five evolutionary algorithms to find the optimal initial solution from the LP. Through simulations, we compared the performance of the evolutionary algorithms including three PSO algorithms and two genetic algorithms, which have difference in the mutation and crossover methods. We conclude that one of the genetic algorithm has a slightly better performance than the others, while all of the five algorithms have fast convergency speed.

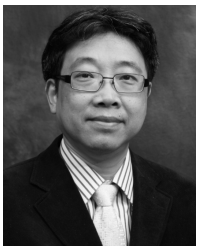
REFERENCES

[1] L. Yang, B. Liu, J. Cao, Y. Sahni, and Z. Wang, "Joint computation partitioning and resource allocation for latency sensitive applications in mobile edge clouds," *IEEE Trans. Services Comput.*, early access, Jan. 1, 2019, doi: 10.1109/TSC.2018.2890603.
[2] L. Yang, J. Cao, Z. Wang, and W. Wu, "Network aware multi-user computation partitioning in mobile edge clouds," in *Proc. 46th Int. Conf. Parallel Process. (ICPP)*, Aug. 2017, pp. 302–311.
[3] C. Esposito, A. Castiglione, F. Pop, and K.-K.-R. Choo, "Challenges of connecting edge and cloud computing: A security and forensic perspective," *IEEE Cloud Comput.*, vol. 4, no. 2, pp. 13–17, Mar. 2017.
[4] Y. Xiong, Y. Sun, L. Xing, and Y. Huang, "Extend cloud to edge with KubeEdge," in *Proc. IEEE/ACM Symp. Edge Comput. (SEC)*, Oct. 2018, pp. 373–377.
[5] S. Wang, J. Xu, N. Zhang, and Y. Liu, "A survey on service migration in mobile edge computing," *IEEE Access*, vol. 6, pp. 23511–23528, 2018.
[6] D. Bernstein, "Containers and cloud: From LXC to docker to kubernetes," *IEEE Cloud Comput.*, vol. 1, no. 3, pp. 81–84, Sep. 2014.
[7] A. Madhavapeddy and D. Scott, "Unikernels: Rise of the virtual library operating system," *Distrib. Comput.*, vol. 11, no. 11, pp. 1–15, 2014.
[8] Y. Ruan, Z. Cao, and Z. Cui, "Pre-Filter-copy: Efficient and self-adaptive live migration of virtual machines," *IEEE Syst. J.*, vol. 10, no. 4, pp. 1459–1469, Dec. 2016.
[9] J. Zhang, F. Ren, and C. Lin, "Delay guaranteed live migration of virtual machines," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2014, pp. 574–582.
[10] U. Deshpande, D. Chan, T.-Y. Guh, J. Edouard, K. Gopalan, and N. Bila, "Agile live migration of virtual machines," in *Proc. IEEE Int. Parallel Distrib. Process. Symp. (IPDPS)*, May 2016, pp. 1061–1070.
[11] S. Nathan, U. Bellur, and P. Kulkarni, "Towards a comprehensive performance model of virtual machine live migration," in *Proc. 6th ACM Symp. Cloud Comput. (SoCC)*, 2015, pp. 288–301.
[12] L. Chaufournier, P. Sharma, F. Le, E. Nahum, P. Shenoy, and D. Towsley, "Fast transparent virtual machine migration in distributed edge clouds," in *Proc. 2nd ACM/IEEE Symp. Edge Comput. (SEC)*, Oct. 2017, pp. 1–13.
[13] A. Machen, S. Wang, K. K. Leung, B. J. Ko, and T. Salonidis, "Live service migration in mobile edge clouds," *IEEE Wireless Commun.*, vol. 25, no. 1, pp. 140–147, Feb. 2018.
[14] Z. Li and G. Wu, "Optimizing VM live migration strategy based on migration time cost modeling," in *Proc. Symp. Archit. Netw. Commun. Syst. (ANCS)*, 2016, pp. 99–109.
[15] L. Ma, S. Yi, and Q. Li, "Efficient service handoff across edge servers via docker container migration," in *Proc. IEEE/ACM SEC*, Oct. 2017, pp. 1–13.
[16] V. Medina and J. M. García, "A survey of migration mechanisms of virtual machines," *ACM Comput. Surveys*, vol. 46, no. 3, pp. 1–33, Jan. 2014.
[17] M. Mishra, A. Das, P. Kulkarni, and A. Sahoo, "Dynamic resource management using virtual machine migrations," *IEEE Commun. Mag.*, vol. 50, no. 9, pp. 34–40, Sep. 2012.
[18] T. G. Rodrigues, K. Suto, H. Nishiyama, and N. Kato, "Hybrid method for minimizing service delay in edge cloud computing through VM migration and transmission power control," *IEEE Trans. Comput.*, vol. 66, no. 5, pp. 810–819, May 2017.
[19] T. G. Rodrigues, K. Suto, H. Nishiyama, N. Kato, and K. Temma, "Cloudlets activation scheme for scalable mobile edge computing with transmission power control and virtual machine migration," *IEEE Trans. Comput.*, vol. 67, no. 9, pp. 1287–1300, Sep. 2018.
[20] K. Ha and Y. Abe and, "Adaptive VM handoff across cloudlets," School Comput. Sci., Carnegie Mellon Univ., Pittsburgh, PA, USA, Tech. Rep. CMU-CS15-113, 2015.
[21] S. Gao and B. Han and, "Solving traveling salesman problem by hybrid particle swarm optimization algorithm," *Control Decis.*, vol. 19, no. 11, pp. 1286–1289, 2004.
[22] M. Patel, S. Chaudhary, and S. Garg, "Performance modeling of skip models for VM migration using xen," in *Proc. Int. Conf. Comput., Commun. Autom. (ICCCA)*, Apr. 2016, pp. 1256–1261.



LEI YANG (Member, IEEE) received the B.Sc. degree from Wuhan University, in 2007, the M.Sc. degree from the Institute of Computing Technology, Chinese Academy of Sciences, in 2010, and the Ph.D. degree from the Department of Computing, The Hong Kong Polytechnic University, in 2014. He has been a Visiting Scholar with Technique University Darmstadt, Germany, from November 2012 to March 2013. He is currently an Associate Professor with the School of Software Engineering, South China University of Technology, China. He has published more than 40 articles in major international conferences proceedings and journals including IEEE INFOCOM, IEEE PERCOM, TMC, ToC, TSC, and so on. His research interests include edge computing, mobile cloud computing, and the Internet of Things with particular focus on task scheduling and resource management.

DOUDOU YANG received the B.Sc. degree from the Nanjing University of Information Science and Technology, China, in 2017. She is currently pursuing the master's degree with the School of Software Engineering, South China University of Technology, China. She is also pursuing the degree with the Laboratory of Mobile Cloud Computing. Her research interests include edge computing, service migration, and cloud computing.



JIANNONG CAO (Fellow, IEEE) received the B.Sc. degree in computer science from Nanjing University, China, in 1982, and the M.Sc. and Ph.D. degrees in computer science from Washington State University, USA, in 1986 and 1990, respectively. He is currently a Chair Professor of distributed and mobile computing with the Department of Computing, The Hong Kong Polytechnic University. He is also the Director of the Internet and Mobile Computing Lab, Department, and also the Director of the University Research Facility in Big Data Analytics. He has coauthored 5 books in mobile computing and wireless sensor networks, co-edited 9 books, and published more than 500 articles in major international journals and conference proceedings. His research interests include parallel and distributed computing, wireless networks and mobile computing, big data and cloud computing, pervasive computing, and fault tolerant computing. He is a member of ACM and a Senior Member of China Computer Federation (CCF).

YUVRAJ SAHNI received the B.E. degree (Hons.) in electrical and electronics engineering from the Birla Institute of Technology and Science, Pilani, India, in 2015. He is currently pursuing the Ph.D. degree with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong. His research interests include wireless sensor networks, middle-ware, and the Internet of Things.

XIAOHUA XU (Member, IEEE) is currently a tenure-track Assistant Professor with the Department of Computer Science, Kennesaw State University. His teaching and research interests include network security, machine learning, and blockchain. Over the years, he has published 57 articles in highly respected conferences and journals, such as the IEEE/ACM TRANSACTIONS ON NETWORKING and IEEE INFOCOM. He has received the Best Paper Award from the Department of Computer Science, Illinois Institute of Technology. His h-index is 18 and his total citation has exceeded 1, 477, according to Google Scholar. Four of his publications have been cited more than 110 times each. He has delivered more than 100 talks at conferences and universities worldwide. He has served on program committees of numerous international conferences and as a Session Chair for IEEE INFOCOM for five times. As a PI, he received an NSF grant totaling $244,808 and two OVPR research grants.