IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received March 1, 2020, accepted March 22, 2020, date of publication March 27, 2020, date of current version April 17, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2983742

A Multi-Objective Task Scheduling Method for Fog Computing in Cyber-Physical-Social Services

MING YANG[1], HAO MA[2], SHUANG WEI[1], YOU ZENG[1], YEFENG CHEN[1], YUEMEI HU[3]

[1]Zhejiang Meteorological Information Network Center, Hangzhou 310016, China
[2]Zhejiang Climate Center, Hangzhou 310016, China
[3]School of Information Science and Technology, Qufu Normal University, Rizhao 276800, China

Corresponding author: Yuemei Hu (yuemeihu@qfnu.edu.cn)

This work was supported in part by the Major Project of Zhejiang Province, China, under Grant 2017C03035, and in part by the Key Project of Zhejiang Meteorological Service under Grant 2017ZD17 and Grant 2018ZD03.

ABSTRACT Fog computing provides users with data storage, computing, and other services by using fog layer devices close to edge devices. Tasks and resource scheduling in fog computing has become a research hotspot. For the multi-objective task-scheduling problem in fog computing, an adaptive multi-objective optimization task scheduling method for fog computing (FOG-AMOSM) is proposed in this paper. In this method, the total execution time and the task resource cost in the fog network are taken as the optimization target of resource allocation, and a multi-objective task scheduling model is designed. Since the objective model is a Pareto optimal solution problem, the global optimal solution can be obtained by using multi-objective optimization theory and the improved multi-objective evolutionary heuristic algorithm. Moreover, to obtain a better distribution of the current task scheduling group, the neighborhood is adaptively changed according to the current situation of the task scheduling group in fog computing, which avoids the problem that the neighborhood value caused by the neighborhood policy in the multi-objective algorithm affects the distribution of the task scheduling population. This algorithm is used to solve the non-inferior solution set of the utility function index of fog computing task scheduling to try to solve the multi-objective cooperative optimization problem in fog computing task scheduling. The results show that the proposed method has better performance than other methods in terms of total task execution time, resource cost and load dimensions.

INDEX TERMS Cloud computing, fog computing, task scheduling, multi-objective optimization algorithm, cyber-physical-social service.

I. INTRODUCTION

With the development of network communication technology, cloud computing as a new type of computing has attracted increasing attention and been widely integrated into various industries. Among them, the internet of things (IoT) provides network infrastructure for cyber-physical-social systems (CPSS) and will play an important role in our daily life. In CPSS, a large number of deployed IoT devices (sensors, actuators, etc.) are connected to collect data related to energy, transportation, urban infrastructure, manufacturing, health care, and public safety and support much of the key infrastructure of smart-world CPSS, such as smart grids, smart transportation [1], smart health, and smart cities [2].

Due to the limited computing and storage capacity of devices connected to the IoT, the explosive growth of terminal devices and the large distance between cloud providers and users, the traditional network architecture of the cloud computing framework has been challenged. For applications such as meteorological element collection terminals, which require real-time performance, a new architecture is needed to quickly respond to the needs of the underlying equipment.

In 2011, Cisco first proposed a new service computing mode – fog computing [3]. Fog computing is a highly virtualized platform, adding a fog layer between the IoT terminal node and the traditional cloud, using the devices in the fog layer to provide elastic computing, storage, network services, and other resources [4]. The fog computing structure is shown in Fig. 1.

The fog computing architecture is a three-layer network structure [5]. The top layer is the cloud computing center layer, which is mainly composed of cloud servers processing and storing a large amount of data; the middle layer is the fog computing layer, which is mainly composed of fog

The associate editor coordinating the review of this manuscript and approving it for publication was Md. Arafatur Rahman.

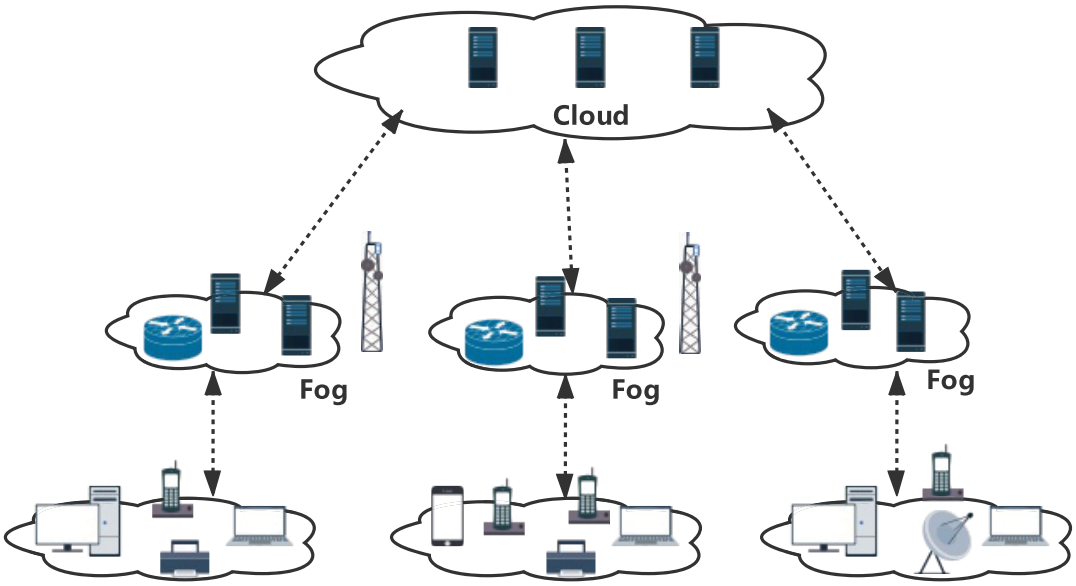


FIGURE 1. Fog computing architecture.

devices with certain computing power and supports mobility, low latency, location awareness, and heterogeneity; and the bottom layer is the IoT device layer of terminal users, which is mainly composed of smartphones and PC machines, weather detection equipment, etc. Fog computing is a processing environment with more widely distributed deployment. Data storage and processing rely more on edge devices, making full use of the resources in the edge network to provide more effective services.

In fog computing, distributed fog resources are used to provide efficient computing services for users through resource allocation. Computing services involve different computing performance, service times, service billing, service energy consumption, etc. There are some computing services that conflict with each other. Therefore, how to coordinate the different objectives between these different entities and within the same entities through a resource management and scheduling algorithm is a hot issue in fog computing. The multi-objective constrained task and resource scheduling problem is a non-deterministic polynomial (NP) problem [6], and it is difficult to obtain optimal solutions.

In this paper, we use the method of the adaptive neighborhood to keep the diversity of the object solution set and try to solve the problem of multi-objective coordination in fog computing task scheduling. On this basis, FOG-AMOSM based on the adaptive neighborhood method is proposed. This algorithm adopts an improved adaptive neighborhood pruning strategy, which can not only prevent degradation of the task and resource scheduling population but also effectively save the extreme-value individuals to ensure the breadth of the Pareto front. In addition, the shortest task completion time and the lowest cost are conflicting goals in general, which prevents obtaining optimal scheduling results at the same time. However, the improved algorithm proposed in this paper can balance the two objectives, and the distribution of computing nodes is better and more uniform. The experimental results show that the proposed method has better performance than other methods in terms of the total task execution time, resource cost and load.

The remainder of this paper is organized as follows. Section II describes the task scheduling model for fog computing in cloud environments. Section III describes task problems and the objective function in fog computing. Section IV presents the methods for adaptive neighborhood multi-objective fog computing task scheduling, including the adaptive neighborhood strategy [7], chromosome coding of multiobjective task scheduling, the main methods of FOG-AMOSM and an overview of the structure of FOG-AMOSM. Section V describes the experimental environment and method used to evaluate and analyze the performance of FOG-AMOSM. In Section VI, related works are summarized. Section VII concludes the paper.

II. MODEL AND DEFINITION

In this section, we introduce the task scheduling model in fog computing and some formalized concepts. The key terms and their descriptions are listed in Table 1.

In a fog computing environment, the task request types of terminal users are divided into time-pressing and nontime-pressing [8]. Time-pressing resource requests are scheduled to a near fog node according to the scheduling strategy to reduce the delay and improve the processing efficiency; non-time-pressing task requests are scheduled to a cloud node according to the scheduling strategy.

When a large number of tasks are sent to the fog server in a short period of time, the fog server will assign a suitable computing service to complete the task. To make more efficient

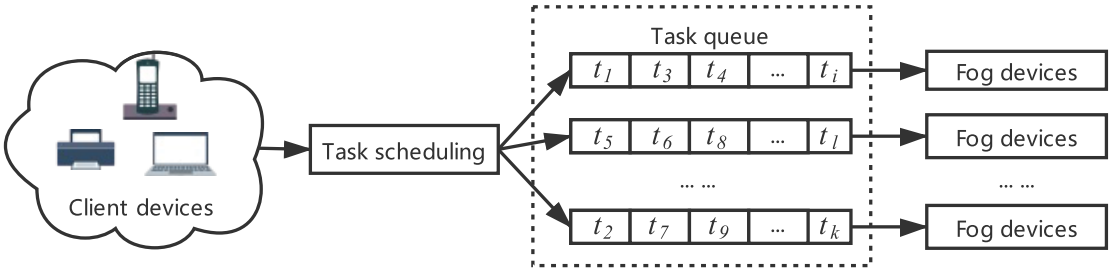


FIGURE 2. Fog computing resource scheduling architecture.

TABLE 1. Key notation definitions.

Notations	Definitions
m	The number of fog computing resouces
n	The number of computing tasks
VM_j	Fog computing resouce j, $1 \leq j \leq m$
T_i	Computing task i, $1 \leq i \leq n$
$Fitness$	The objective function of task scheduling
T_{total}	The computing time of all tasks
C_{total}	The service cost of all tasks
ET_{ij}	The computing time matrix of task T_i on resource VM_j
CT_{ij}	The expected completion time of task T_i on resource VM_j
b_j	The earliest available time of processing node VM_j
C_{ij}	The cost matrix of task T_i on resource VM_j
u_j	The price of computing resource node VM_j
V	The total penalty cost of violation
P	The resource scheduling population set
N	The number of resource-scheduling population
$NDSet$	The non-dominated task scheduling set

use of computing resources and meet the user's requirements, it is very important to reasonably schedule the tasks.

Fig. 2 shows the task scheduling model structure for the fog computing platform, in which the computing resources in the fog service center are computing nodes, which will process computing tasks with different capacities. In this structure chart, the user request is the computing task submitted by the user, which is expected to be processed on the computing resources followed by return of the computing results. Task scheduling is based on the comprehensive consideration of efficiency, energy consumption, etc.; it allocates tasks to the corresponding task queue of the corresponding computing resources and waits for processing. The task scheduling here will be the focus of this paper.

For study convenience, it is assumed that the data transmission in the process of resource scheduling and computation result return is not limited by the communication bandwidth, and the global communication consumption is relatively stable. In this paper, the computing power of fog computing nodes is taken as the main measurement standard to ensure that the computing resources available to the computing service equipment can meet the deadline requirements of the task, prevent task overtime and reduce the total completion delay of the task. Furthermore, the revenue mechanism of fog computing is introduced to ensure that the task budget completes resource allocation on the premise of meeting the minimum income needs of service providers. The related definitions of fog computing resource scheduling discussed in this paper are as follows:

1. Suppose that $VM_{fog} = \{VM_1, VM_2, \cdots, VM_m\}$, where VM_{fog} is the fog computing resource nodes, and VM_j is fog computing resource j, $1 \leq j \leq m$.

2. Suppose that $T = \{t_1, t_2, \cdots, t_n\}$, where n is the number of computing tasks, and t_i is task i, $1 \leq i \leq n$.

3. The computing time of tasks is represented by matrix ET of size $m \times n$. ET_{ij} is the expected execution time of task t_i on computing node VM_j, and $ET_{ij} = 0$ indicates that task t_i cannot be executed on computing node VM_j.

4. The service cost of computing tasks is represented by matrix C of size $m \times n$. C_{ij} is the cost of task t_i on computing node VM_j, and u_j is the price of resource VM_j in unit time.

III. PROBLEM DESCRIPTION

In the fog computing architecture, from the perspective of terminal users, they always hope that the allocated resources can meet the task and resource requirements as much as possible and complete all submitted tasks with minimum execution time and minimum cost to the service providers. However, from the perspective of fog computing resources, the same task set being completed in less time requires higher performance of the fog computing resources and a faster computing speed. As a result, the higher performance of the fog computing resources and the faster computing speed contribute to a higher cost of the computing tasks. Furthermore, higher availability of the fog server and a greater number of requests also lead to a higher price of the fog computing resources. Therefore, the cost of resources plays a decisive role in the cost of terminal users and is one of the important indicators in the evaluation model.

In addition, task scheduling in a fog computing architecture often takes efficiency as the primary goal. The current processing capacity is no longer a problem under the support of hardware conditions. The key problem is that a large number of processing machine operations in the resource center will consume a large amount of electric energy, which not only wastes energy and increases costs but also causes the problem of heat dissipation in the center. If the center completes the submitted tasks in the fastest time, then the heat accumulated by the center will rapidly increase in unit time, which will

incur a higher cost to complete the heat dissipation work. The following describes the total execution time and the cost of fog computing resource node task scheduling.

A. TOTAL EXECUTION TIME

Assume that the cross-node execution of computing tasks is not considered, the task submitted by the terminal user is the minimum unit of task allocation by the scheduler, the decomposition of large computing tasks is completed by the terminal user, and the tasks are independent, without communication and data synchronization, thus avoiding the performance degradation caused by frequent communication between tasks. According to the assumptions that tasks in the task set are independent and these tasks are not decomposable, the set is called a meta task set.

It is assumed that n tasks $T = \{t_1, t_2, \cdots, t_n\}$ are scheduled to m fog computing resource nodes $VM_{fog} = \{VM_1, VM_2, \cdots, VM_m\}$ in a reasonable manner to achieve the expected goal as much as possible. The result of task scheduling is represented by a two-dimensional matrix X of size $n \times m$. $x_{ij} = 1\left(x_{ij} \in X\right)$ indicates that task t_i will be scheduled to computing node VM_j for execution; otherwise, $x_{ij} = 0$.

For description, the following parameters and symbols are defined: T_{total} is the total computing time, a_i is the arrival time of task t_i, b_j is the earliest available time of computing node VM_j, ET_{ij} is the expected execution time of task t_i on computing node VM_j, and CT_{ij} is the expected completion time of task t_i on computing node VM_j. CT_{ij} is calculated by

$$CT_{ij} = b_j + ET_{ij} \tag{1}$$

Generally, task t_i is scheduled to be executed on computing node VM_j. Suppose that $CT_i = CT_{ij}$ for t_i indicates the total time of task execution; then, the execution completion time is $\max_{t_i \in T}(CT_i)$ for task set T. The total time of task execution is calculated by

$$T_{total} = \sum_{i=1}^{n}\sum_{j=1}^{m} CT_{ij} \tag{2}$$

B. COST OF RESOURCE NODES IN FOG COMPUTING

Commonly, a shorter task computing time submitted by the terminal user requires more fog computing resources. Similarly, a higher CPU processing power of computing resources means a higher resource cost. Therefore, how to minimize the cost and total computing time of fog computing resources is a complex problem.

SLA is a level agreement on the service quality determined by the service provider and user through negotiation, mainly covering service performance, service time, service billing and other measurable service quality characteristics [9]. The penalty of service level objective (SLO) [10] is the specific measurable characteristics of SLA, such as the availability, throughput, frequency, response time, and quality. SLO provides a quantitative means to define the service level that customers can obtain from the provider.

The cost problem mainly includes SLO violation and the cost of computing resource node processing terminal requests.

(1) SLO violation penalty function: if V is the total penalty cost of SLO violation, then the value of the violation penalty function is measured by

$$V = \sum_{i}^{q}[\gamma_i + \eta \times \left(t_i^{stop} - t_i^{deadline}\right)] \tag{3}$$

where i is the sequence number of requests with SLO violation due to service timeout, q is the total number of requests with SLO violation, γ_i is the basic penalty cost of request with SLO violation i, η is the penalty cost of unit time caused by timeout, and t_i^{stop} is the actual response time of request with SLO violation i.

(2) Cost function of fog computing resource nodes: assuming that C_{total} is the total cost required to complete all tasks, u_j is the price of resource VM_j in unit time, and C_{ij} is the cost of task t_i on resource VM_j, then the execution cost C_{ij} of task t_i can be expressed as

$$C_{ij} = ET_{ij} \times u_j \tag{4}$$

$$C_{total} = \sum_{i=1}^{n}\sum_{j=1}^{m} C_{ij} + V \tag{5}$$

C. OBJECTIVE FUNCTION

Based on the above model of total execution time and cost, the objective function of the adaptive neighborhood multi-objective fog computing task scheduling algorithm model can be expressed as

$$\begin{cases} \min(T_{total}) = \min(\sum_{i=1}^{n}\sum_{j=1}^{m}(b_j + ET_{ij})) \\ \min(C_{total}) = \min(\sum_{i=1}^{n}\sum_{j=1}^{m}(ET_{ij} \times U_j)) + V \end{cases} \tag{6}$$

where ET_{ij} is the expected execution time of task t_i on computing node VM_j, m represents the number of computing resources, n represents the task number, U_j is the price of resource VM_j in unit time, and min is the minimum value function.

From formula 6, we can obtain the definition of the objective function $Fitness(Z_1, Z_2)$, which can be calculated by

$$Fitness(Z_1, Z_2) = \begin{cases} Z_1 = \min(T_{total}) \\ Z_2 = \min(C_{total}) \end{cases} \tag{7}$$

where Z_1 is the objective function of the total time required to complete all tasks, Z_2 is the objective function of the total cost required to complete all tasks, and b_j is the earliest available time of processing node VM_j.

Formula 7 shows that the goal of task scheduling is to achieve the minimum total execution time and the minimum total cost after all tasks are executed to achieve balance of the goals.

IV. METHODS FOR ADAPTIVE MULTI-OBJECTIVE TASK SCHEDULING

According to the definition of the multi-objective problem for fog computing task scheduling in the previous section, the following details the design of each part of the adaptive neighborhood multi-objective fog computing task scheduling algorithm.

In fog computing, task scheduling is an NP problem. For this type of problem, a heuristic algorithm can often find a relatively satisfactory solution, and the results of multi-objective problems can be expressed as Pareto-optimal solutions, which provide different solutions. The multi-objective task scheduling method adopted in this paper is to find the optimal solution from the Pareto solutions according to the needs of task and resource scheduling.

A. ADAPTIVE NEIGHBORHOOD STRATEGY

To avoid the phenomenon that the distribution of the algorithm is destroyed in the evolution and to solve the problem of the neighborhood radius value, first, we should perform adaptive adjustment of the radius change rate of the sigmoid function curve slowly under the condition of the average distance to improve the neighborhood radius of the individual whose distance is close to the average distance. Second, when the difference between the average distance and the minimum distance is large, we should ensure that the adaptive adjustment curve does not tend to a straight line. Finally, the better individuals in the population still have a certain distribution. At the same time, to maintain a better distribution as much as possible, the adaptive adjustment curve at the average distance should be smoothed.

The sigmoid function is used as the adaptive adjustment curve, which is measured by

$$f(x) = \frac{1}{1 + \exp(-ax)} \tag{8}$$

where a is a constant value. As shown in formula 8, this function exhibits a nice balance between linear and nonlinear behaviors.

The self-adaptive adjustment formula of the radius change rate when solving the value problem of the neighborhood radius designed by the sigmoid function is shown in formula 9. In formula 9, let $r^{(t-1)}$ be the neighborhood radius of the previous generation to judge whether the value is greater than the minimum spacing between current individuals ${d_{min}}^{(t)}$, and then, let $r^{(t)}$ be the neighborhood radius of the current generation, which is calculated by

$$r^{(t)} = \begin{cases} \dfrac{r_{\max} - r_{\min}}{1 + \exp\left(\dfrac{a\left(r^{(t-1)} - {d_{\min}}^{(t)}\right)}{{d_{avg}}^{(t)} - {d_{\min}}^{(t)}}\right)} + r_{\min} \\ r_{\min}, \; r^{(t-1)} < d_{\min}, \end{cases} \qquad r^{(t-1)} \geq {d_{\min}}^{(t)} \tag{9}$$

where T is the largest evolutionary algebra, $t = 1, 2, \cdots, T$ is the current evolutionary algebra, a is any constant, r_{max} is the maximum neighborhood radius, r_{min} is the minimum neighborhood radius, ${d_{min}}^{(t)}$ is the minimum spacing between current individuals, and ${d_{avg}}^{(t)}$ is the average spacing between current individuals.

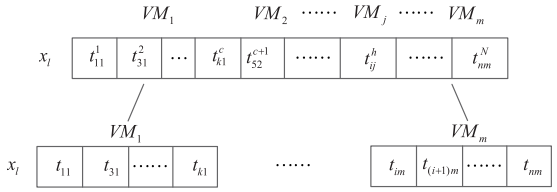


FIGURE 3. Chromosome encoding structure for task scheduling in fog computing.

In formula 8 and formula 9, we can see that the change rate of the neighborhood radius is non-linearly adjusted with the sigmoid curve between the minimum distance and the average distance according to the distance between individuals. Obviously, when most of the individuals in the population have a similar distance and the minimum distance is close to the average distance, the distribution of the population is improved.

B. MULTI-OBJECTIVE CHROMOSOME ENCODING

For the resource scheduling problems in fog computing, FOG-AMOSM is mainly based on the population to solve the multi-objective problem of resource scheduling. Therefore, the encoding of population chromosomes is very important. The traditional multi-objective algorithm encodes the alternatives of the decision space into a string of 0 and 1. For the task scheduling problem of fog computing resources, it is difficult to use this coding method because the coding method not only represents the assignment of tasks (that is, which fog computing resource node a job is assigned to) but also represents the scheduling information of its computing resources to jobs (that is, the execution order of jobs on a computing resource). In this way, it is difficult to include all the information of the mapping in binary encoding.

In this paper, we use real coding to encode task-scheduling chromosomes. Fig. 3 shows a chromosome encoding structure for task scheduling in fog computing. As indicated in Fig. 3, the chromosome (i.e., scheduling x_l scheme) is encoded into a multidimensional matrix, and each gene in the chromosome represents the resource scheduling strategy of computing task t_i. A chromosome x_l consists of a set of genes, which represents a set of resource scheduling strategies for the computing task. Meanwhile, a chromosome x_l also records the execution order of the task set in the resource queue. The set $X = \{x_1, x_2, \cdots, x_N\}$ of all chromosomes is a population.

In Fig. 3, x_l is the l-th chromosome, VM_j is the j-th fog computing resource node, t_{ij}^h is the i-th task assigned to the j-th fog computing resource node VM_j, h is the h-th chromosome, and N is the length of the individual (i.e., scheduling scheme).

Algorithm 1 Algorithm of the Initial Resource Scheduling Population

Require: VM, T, m, n, N
Ensure: Resource scheduling schemes X
1: **for** $i = 1$ to N **do**
2: **for** $j = 1$ to n **do**
3: Randomly select a task t_j and assign it to computing resource vm_k, $vm_k \in VM, 0 \leq k \leq m, t_j \in T$
4: Generate a task scheduling scheme x_i
5: **end for**
6: **end for**
7: Generate a new initial population of scheduling scheme X
8: **return** X

C. MAIN METHODS OF FOG-AMOSM

1) ALGORITHM FOR THE INITIAL RESOURCE SCHEDULING POPULATION

The initial population consists of N randomly generated scheduling schemes. The resource scheduling algorithm of random population generation is shown in Algorithm 1.

2) CALCULATION OF THE DISTRIBUTION FITNESS OF SCHEDULING INDIVIDUALS

The evaluation of the distribution fitness mainly involves screen the chromosomes of the task scheduling archive set through the objective function and then further screening the density value, distance value and neighborhood relationship matrix of the selected scheduling scheme chromosomes through the objective function of task scheduling. The algorithm for calculating the chromosome distribution fitness of a scheduling scheme is shown in Algorithm 2.

3) RESOURCE SCHEDULING POPULATION MAINTENANCE

To provide diversity and distribution of the resource scheduling population, according to the density value, distance value and neighborhood relationship of individuals in the population, population maintenance is carried out through selection, crossover, variation and other processes. The population maintenance algorithm for calculating the chromosome of the scheduling method is shown in Algorithm 3.

D. METHOD OVERVIEW OF FOG-AMOSM

According to the improved methods presented in the previous section, the implementation process of FOG-AMOSM is as follows:

Step 1 task scheduling coding: according to the task scheduling coding scheme, the real number coding method is adopted, and the upper and lower limits of the value are determined by the numbers of resources and tasks.

Step 2 generation of the initial task scheduling population: randomly generate an initial task scheduling population $P(t)$, leave the task scheduling archive set $Q(t)$ empty, and set the evolution algebra $t = 0$, where the sizes of $P(t)$ and $Q(t)$ are equal to the number of tasks N.

Algorithm 2 Calculation of the Chromosome Distribution Fitness of a Scheduling Scheme

Require: Current task scheduling set $Q(t)$
Ensure: The matrix of density value $\gamma(t)$, distance value $\eta(t)$ and neighborhood relation in individuals set R
1: $c = 0$
2: **while** not end of $Q(t)$ **do**
3: $c = c + 1$
4: **end while**
5: **for** $i = 1$ to m **do**
6: $\gamma_i(t) = 0$
7: $\eta_i(t) = 0$
8: **for** $j = 1$ to n **do**
9: **if** $(i \neq j$ and $|Q[i], Q[j]| < r)$ **then**
10: Calculate $\gamma_{i,j}(t)$ by Eq.(7)
11: Calculate $\eta_{i,j}(t)$ by Eq.(9)
12: Record the neighborhood relationship between individuals: $R_{i,j}(t)$
13: **end if**
14: **end for**
15: **end for**
16: **return** $R(t)$

Step 3 evolutionary operation: use the binary tournament rule to select the resource individuals in $P(t)$ to enter the mating pool; then, perform crossover and mutation operations on the resource individuals in the mating pool, and assign the new resource individuals to $Q(t)$.

Step 4 evaluate fitness of individual: use $VM(t) = P(t) \cup Q(t)$, where $R(t)$ is twice the number of tasks, i.e., $2N$, and then, use the objective function defined in formula 7 to evaluate the fitness of all individuals in $VM(t)$.

Step 5 elite strategy: save all the non-dominated individuals in $VM(t)$ to $NDSet$. If $|NDSet| > N$, use the adaptive neighborhood algorithm to prune and reduce the size of this set until its size equals the number of tasks N. If $|NDSet| < N$, the dominant individual is selected from $VM(t)$ to fill up this set until its size equals the number of tasks N.

Step 6 termination condition: generally, the evolutionary algebra is the termination condition. If the termination condition is met, then all the non-dominated individuals in $NDSet$ will be regarded as the final output group. Otherwise, save $NDSet$ to $P(t + 1)$, set $t = t + 1$, and go to Step 3.

V. EXPERIMENT AND ANALYSIS

In this section, we evaluate the performance optimization of our method. Specifically, we first introduce simulation setup including experimental environment and parameter setting. Then, we evaluate the performance of the task scale in terms of total computing time and service costs. Finally, we analyze the evaluation results.

Algorithm 3 Algorithm of Resource Scheduling Population Maintenance

Require: Current task scheduling population $P(t)$, non-dominated task scheduling set $NDSet$, the neighborhood relationship $R(t)$

Ensure: Optimized task scheduling population $O(t)$

$N = 0$
while not end of $P(t)$ **do**
  $N = N + 1$
end while
Delete index set $D = \emptyset$
$i = 1$
while $i > N$ **do**
  Sort in descending order according to the neighborhood density $\gamma(t)$
  if $(\gamma_{i-1}(t) \neq \gamma_i(t))$ **then**
    $D = D \cup Max(\gamma_{i-1}(t), \gamma_i(t))$
  else if $(\gamma_{i-1}(t) == \gamma_i(t)$ and $\eta_{i-1}(t) \neq \eta_i(t))$ **then**
    $D = D \cup Min(\eta_{i-1}(t), \eta_i(t))$
  end if
  for $j = 1$ to n **do**
    if $(D_j, P_j(t) \in R(t)$ **then**
      $\gamma_j(t) = \gamma_j(t) - 1$
      $\eta_j(t) = \eta_j(t) - D_j$
    end if
  end for
  $i = i + 1$
end while $NDSet = P(t) - D$
return $NDSet$

TABLE 2. Parameters of VM resources in the fog computing environment.

Number	Parameter	Value
1	CPU Processing Capacity	[800,1500] MIPs
2	CPU Core Number	[1,4]
3	Bandwidth	[100,1000] M/s
4	RAM	4 GB
5	Storage	[500,1000] GB

TABLE 3. Parameters of tasks in the fog computing environment.

Number	Parameter	Value
1	Task Length	[500,20000] MIPs
2	Task Number	[20,160]

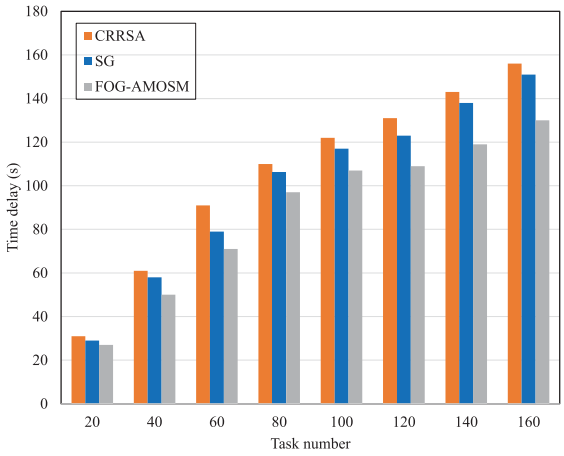


FIGURE 4. Time delay comparison for different task request numbers.

A. EXPERIMENTAL ENVIRONMENT AND PARAMETER SETTING

In our simulation, Cloudsim 5.0 [11] is employed to simulate the fog computing simulation environment. The CPU of the experiment processor is an Intel Core i5 @ 2.30 GHz with 4GB of memory, the operation system is Windows 10 64-bit, and the development tools are Eclipse and JDK 1.8.0.

Fog computing environments support a variety of heterogeneous devices to create fog nodes. In this paper, the diversity of fog nodes and the computing power of different types of fog nodes are considered. In the experiment, the CPU computing power of fog computing nodes ranges from 800 to 1500 MIPs, the bandwidth ranges from 100 to 1000 MB, the computing cost ranges from 3 to 6, and the number of resource nodes is set to 6. The randomly generated terminal-user task has a task length ranging from 500 to 20000 MIPs, the number of tasks ranges from 20 to 160, the basic penalty for violation is set to 3, and the unit time penalty is set to 5. The VM resource parameters in the fog computing environment are listed in Table 2. The task parameters are reported in Table 3.

FOG-AMOSM uses real number coding, the cross probability is 0.87, the mutation probability is 0.012, the evolution algebra is 320, the number of fitness evaluation times is 25000, and the operation algebra of the algorithm is the number of evaluation individuals divided by the population size.

B. EXPERIMENTAL RESULTS AND PERFORMANCE ANALYSIS

In the fog computing model, the simulation experiment is carried out by comparing our method with other methods, including the classical round-robin scheduling algorithm (CRRSA) and the simple genetic algorithm (SGA) [12] for task scheduling. SGA is a heuristic optimization algorithm based on the simulation of genetic mechanism and the dynamic process of biological genetic evolution population, which is used to complete the adaptive allocation of computing tasks and computing resources. The experiment is divided into three parts: the first part is an experiment comparing the delay and execution times; the second part is an experiment comparing the violation rates and service costs; and finally, the load diversities of the algorithms are compared and analyzed.

1) TIME DELAY AND TOTAL EXECUTION TIME

The delay and execution times of each algorithm are compared with other methods by setting different numbers of terminal tasks. In Figs. 4 and 5, the delay and execution times

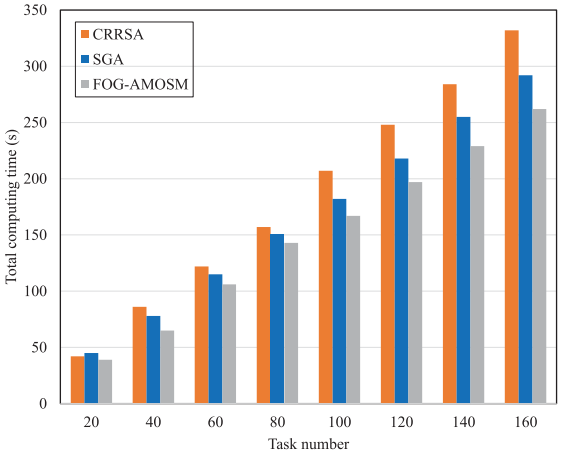


FIGURE 5. Total computing time comparison for different task request numbers.

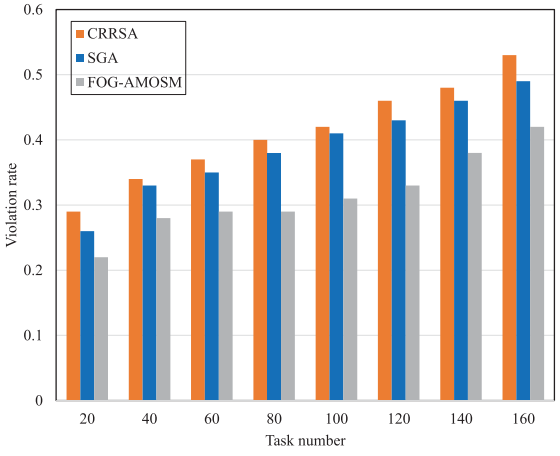


FIGURE 6. Violation rate comparison for different task request numbers.

(time unit: second), respectively, obtained by using the different scheduling algorithms are shown. From Figs. 4 and 5, it can be seen that the FOG-AMOSM task scheduling algorithm is more efficient than the other basic scheduling algorithms in terms of the delay and execution times. In Fig. 4, when the number of tasks range from 20 to 100, the advantages of the FOG-AMOSM method are reduced, mainly because the size of the fog node set is small, and the performance of each round of resource allocation is limited by the number of current fog computing resource nodes. With increasing number of fog computing resource nodes, this problem can be improved. In Fig.5, when the number of tasks is lower, the total computing time of FOG-AMOSM is approximately equal to the CRRSA method. With increasing number of tasks, the FOG-AMOSM method shows a great advantage. Because FOG-AMOSM maintains a good distribution and uses the evaluation function to assess the performance of the solution in each solution set, so when the solution set is large, the advantage can be better reflected.

2) VIOLATION RATE AND SERVICE COST

In this subsection, we focus on evaluating the violation rates and service costs of our method and making comparison with multiple methods by setting different numbers of computing tasks. The experimental results are shown in Figs. 6 and 7. As can be seen from Fig. 6, the overall violation rate of FOG-AMOM is lower than that of CRRSA and SGA, mainly because the optimization of multiple objectives is considered in the FOG-AMOSM algorithm, which has better performance in obtaining the optimal solution, effectively shortens the response time, and thus reduces the violation rate of SLO. However, when the number of tasks reaches 140, the violation rate increases, which may be due to the large number of localized task requests, while the processing capacity of the fog computing resource node is insufficient, resulting in some task requests not being completed within the specified time. It can be seen from Fig. 7 that the overall service cost of

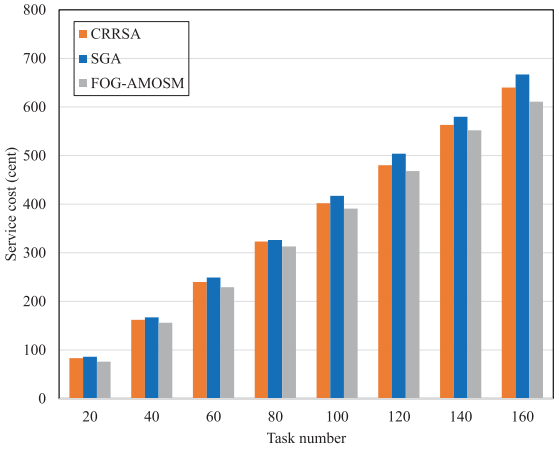


FIGURE 7. Total service cost comparison for different task request numbers.

FOG-AMOSM is also less than other comparison methods. The main reason is that the overall violation rate of the allocation strategy in this paper is low, which leads to a decrease in the service cost, thus effectively reducing the overall service cost.

3) LOAD BALANCING PERFORMANCE

To further study the performance of the scheme in this section, based on the first part of the experiment, we choose 100 tasks and 8 VM nodes of fog computing resources. It can be seen from Fig. 8 that FOG-AMOSM has a greater improvement in load balancing performance than the SGA, mainly due to the following reasons: FOG-AMOSM has the ability to solve the optimal solution of multiple task scheduling objectives, and can balance the load to a certain extent. Among them, the loads of each fog computing resource in the SGA algorithm are extremely unbalanced. The main reason is that the objective function of the SGA algorithm adopts a single objective, and its distribution is poor. The task with the shortest execution time is allocated to the resource computing

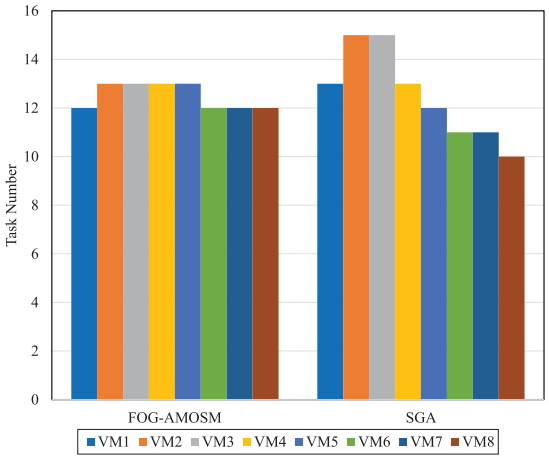


FIGURE 8. Load balancing performance of resource task-scheduling in fog computing.

node with the smallest load, which results in the task with the longest execution time being allocated to a resource processing node with a larger load.

To summarize, the FOG-AMOSM algorithm performs well in different fog computing situations. When FOG-AMOSM processes the two task scheduling goals of execution time and service cost, it can make the two goals reach the optimal values at the same time. The experimental results show that the FOG-AMOSM method has a great advantage in terms of total computing time and service cost. In the worst case, FOG-AMOSM is approximately similar to the CRRSA method, which has advantages in the case of small number of tasks. However, with increasing number of tasks, the FOG-AMOSM method is superior to other similar methods, which verifies the effectiveness of FOG-AMOSM.

VI. RELATED WORK

In recent years, with the development of the IoT, the IoT will play an important role in CPSS [13], [14] [15], [16]. In addition, with the rapid increase in the number of network edge devices, the massive amount of data generated by these network edge devices requires better cloud computing and big data methods.

There are a large number of works that present resource task scheduling methods for fog computing. In [17], Song *et al.* presented a fog computing dynamic load balancing mechanism based on graph repartitioning. Xie and Lyu [18] designed a fog computing task scheduling strategy based on an improved genetic algorithm. In [19], BITAM *et al.* designed a fog computing job scheduling optimization algorithm based on the bee swarm algorithm. Zeng *et al.* [20] described joint optimization of task scheduling and image placement in fog computing supported software-defined embedded system. In other respects, many resource task scheduling methods for cloud computing have been proposed [21], [22] [23], [24] [25], [26].

At the moment, with the development of edge computing, edge computing has become a common technology in cloud computing environments. Xu *et al.* [27] designed an edge computing-enabled computation offloading method with privacy preservation for the internet of connected vehicles. Wang *et al.* [28] described a cloud-edge computing framework for cyber-physical-social services. In [29] Xu *et al.* presented a computation offloading method for big data in IoT-enabled cloud-edge computing. In [30], a blockchain-enabled computation offloading method was proposed. Reference [31] described joint optimization of the offloading utility and privacy for the edge computing-enabled IoT. Cao *et al.* [32] described intelligent offloading in multi-access edge computing.

In order to process large volume of data collected by the IoT and sensor networks in real time. efficient big data storage and recommendation technologies are needed in cloud computing. In [33], Yang *et al.* presented an efficient storage and service method for multi-source merging of meteorological big data in the cloud environment. Wang *et al.* [34] described a tensor-based big data-driven routing recommendation approach for heterogeneous networks. Xu *et al.* [35] described dynamic resource provisioning with fault tolerance for data-intensive meteorological workflows in the cloud.

In addition, there is also some research related to technologies [36], [37] [38], [39] [40], such as mobile services [41], [42], wireless networks [43], [44], and optimization and integration algorithms [45], which will contribute to the efficiency of future processing in cloud computing.

To the best of our knowledge, there are few studies on resource task scheduling methods for fog computing that aim to realize effective computing and quick computing in real time for edge devices in cloud environments.

VII. CONCLUSION AND FUTURE WORK

With the development of cloud computing technology, fog computing has gradually become an important middle layer in cloud computing, which has a stronger processing capacity and a lower time delay than terminal equipment. The task scheduling problem in fog computing is an important problem in fog computing, and its calculation method will directly affect the efficiency and results of task execution in fog environments. In addition, the traditional single-objective task scheduling method cannot effectively meet the multi-objective requirements of the individual task request in the fog computing environment, so it is necessary to have a reasonable management mode and a multi-objective scheduling algorithm to fully utilize the characteristics of fog computing and meet the multi-objective requirements.

Through the research of this paper, the following three conclusions are obtained. First, according to the current situation of the population itself, a multi-objective evolutionary algorithm based on the adaptive neighborhood method is designed. The adaptive neighborhood strategy adopted by the algorithm can effectively solve the optimization problems and maintain a better distribution of the population.

Second, according to the characteristics of fog computing task scheduling, an improved multi-objective task scheduling model for fog computing, FOG-AMOSM, is proposed to optimize the performance and resource cost. Through the definition of the fog computing task scheduling problem, two conflicting object functions of fog computing task scheduling are obtained. Meanwhile, the specific design and implementation process of each part of the model are discussed.

Finally, the performance of FOG-AMOSM is analyzed by comparing with other methods. The experimental results show that FOG-AMOSM is effective at solving the multi-objective optimization problem of fog computing task scheduling.

In the future, we will adjust our method to implement the task scheduling in physical environment. Moreover, additional studies to understand more completely the key strategies of resource scheduling in fog computing are required.

REFERENCES

[1] M. A. Rahman, M. Y. Mukta, A. Yousuf, A. T. Asyhari, M. Z. A. Bhuiyan, and C. Y. Yaakub, "IoT based hybrid green energy driven highway lighting system," in *Proc. IEEE DASC*, Aug. 2019, pp. 587–594.

[2] H. Song, R. Srinivasan, T. Sookoor, and S. Jeschke, *Smart Cities: Foundations, Principles, and Applications*. Hoboken, NJ, USA: Wiley, 2017, pp. 1–906.

[3] F. Bonomi, "Connected vehicles, the Internet of Things, and fog computing," in *Proc. 8th ACM Int. Workshop Veh. Inter-Netw. (VANET)*, Las Vegas, NV, USA, 2011, pp. 13–15.

[4] M. Aazam and E.-N. Huh, "Fog computing: The cloud-IoT/IoE middleware paradigm," *IEEE Potentials*, vol. 35, no. 3, pp. 40–44, May/Jun. 2016.

[5] S. Sarkar and S. Misra, "Theoretical modelling of fog computing: A green computing paradigm to support IoT applications," *IET Netw.*, vol. 5, no. 2, pp. 23–29, Mar. 2016.

[6] H. Topcuoglu, S. Hariri, and M.-Y. Wu, "Performance-effective and low-complexity task scheduling for heterogeneous computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 13, no. 3, pp. 260–274, Mar. 2002.

[7] S. J. Xue and M. Yang, "Improved multi-objective evolutionary algorithm based on adaptive neighborhood," *Comput. Eng. Appl.*, vol. 47, no. 25, pp. 49–53, 2011.

[8] M. Chiang and T. Zhang, "Fog and IoT: An overview of research opportunities," *IEEE Internet Things J.*, vol. 3, no. 6, pp. 854–864, Dec. 2016.

[9] L. Wu, S. K. Garg, S. Versteeg, and R. Buyya, "SLA-based resource provisioning for hosted software-as-a-service applications in cloud computing environments," *IEEE Trans. Services Comput.*, vol. 7, no. 3, pp. 465–485, Jul. 2014.

[10] K.-K. Han, Z.-P. Xie, and X. Lv, "Fog computing task scheduling strategy based on improved genetic algorithm," *Comput. Sci.*, vol. 45, no. 4, pp. 143–148, 2019.

[11] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, "CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Softw., Pract. Exper.*, vol. 41, no. 1, pp. 23–50, Jan. 2011.

[12] M. D. Vose, "The simple genetic algorithm," *Evol. Comput.*, vol. 3, no. 4, pp. 453–472, 1995.

[13] H. Song, G. A. Fink, and S. Jeschke, *Security and Privacy in Cyber-Physical Systems: Foundations, Principles and Applications*. Chichester, U.K.: Wiley, 2017, pp. 1–472.

[14] Y. Jiang, H. Song, Y. Yang, H. Liu, M. Gu, Y. Guan, J. Sun, and L. Sha, "Dependable model-driven development of CPS: From stateflow simulation to verified implementation," *ACM Trans. Cyber-Phys. Syst.*, vol. 3, no. 1, pp. 1–31, Aug. 2018, doi: 10.1145/3078623.

[15] X. Wang, L. T. Yang, Y. Wang, X. Liu, Q. Zhang, and M. J. Deen, "A distributed tensor-train decomposition method for cyber-physical-social services," *ACM Trans. Cyber-Phys. Syst.*, vol. 3, no. 4, pp. 1–15, Oct. 2019, doi: 10.1145/3323926.

[16] X. Wang, W. Wang, L. T. Yang, S. Liao, D. Yin, and M. J. Deen, "A distributed HOSVD method with its incremental computation for big data in cyber-physical-social systems," *IEEE Trans. Comput. Social Syst.*, vol. 5, no. 2, pp. 481–492, Jun. 2018.

[17] S. Ningning, G. Chao, A. Xingshuo, and Z. Qiang, "Fog computing dynamic load balancing mechanism based on graph repartitioning," *China Commun.*, vol. 13, no. 3, pp. 156–164, Mar. 2016.

[18] Z. P. Xie and X. Lyu, "A fog computing task scheduling strategy based on improved genetic algorithm," *Comput. Sci.*, vol. 45, no. 4, pp. 137–142, 2018.

[19] S. Bitam, S. Zeadally, and A. Mellouk, "Fog computing job scheduling optimization based on bees swarm," *Comput. Sci.*, vol. 45, no. 4, pp. 137–142, 2018.

[20] D. Zeng, L. Gu, S. Guo, Z. Cheng, and S. Yu, "Joint optimization of task scheduling and image placement in fog computing supported software-defined embedded system," *IEEE Trans. Comput.*, vol. 65, no. 12, pp. 3702–3712, Dec. 2016.

[21] M. B. Gawali and S. K. Shinde, "Task scheduling and resource allocation in cloud computing using a heuristic approach," *J. Cloud Comput.*, vol. 7, no. 1, p. 4, Dec. 2018.

[22] F. Ebadifard and S. M. Babamir, "A PSO-based task scheduling algorithm improved using a load-balancing technique for the cloud computing environment," *Concurrency Comput., Pract. Exper.*, vol. 30, no. 12, p. e4368, Jun. 2018.

[23] L. Qi, Y. Chen, Y. Yuan, S. Fu, X. Zhang, and X. Xu, "A QoS-aware virtual machine scheduling method for energy conservation in cloud-based cyber-physical systems," *World Wide Web*, vol. 23, no. 2, pp. 1275–1297, Mar. 2020.

[24] X. Xu, S. Fu, L. Qi, X. Zhang, Q. Liu, Q. He, and S. Li, "An IoT-oriented data placement method with privacy preservation in cloud environment," *J. Netw. Comput. Appl.*, vol. 124, pp. 148–157, Dec. 2018.

[25] X. Yang, Z. Chen, K. Li, Y. Sun, N. Liu, W. Xie, and Y. Zhao, "Communication-constrained mobile edge computing systems for wireless virtual reality: Scheduling and tradeoff," *IEEE Access*, vol. 6, pp. 16665–16677, 2018.

[26] L. Zuo, L. Shu, S. Dong, C. Zhu, and T. Hara, "A multi-objective optimization scheduling method based on the ant colony algorithm in cloud computing," *IEEE Access*, vol. 3, pp. 2687–2699, 2015.

[27] X. Xu, Y. Xue, L. Qi, Y. Yuan, X. Zhang, T. Umer, and S. Wan, "An edge computing-enabled computation offloading method with privacy preservation for Internet of connected vehicles," *Future Gener. Comput. Syst.*, vol. 96, pp. 89–100, Jul. 2019.

[28] X. Wang, L. T. Yang, X. Xie, J. Jin, and M. J. Deen, "A cloud-edge computing framework for cyber-physical-social services," *IEEE Commun. Mag.*, vol. 55, no. 11, pp. 80–85, Nov. 2017.

[29] X. Xu, Q. Liu, Y. Luo, K. Peng, X. Zhang, S. Meng, and L. Qi, "A computation offloading method over big data for IoT-enabled cloud-edge computing," *Future Gener. Comput. Syst.*, vol. 95, pp. 522–533, Jun. 2019.

[30] X. Xu, Y. Li, T. Huang, Y. Xue, K. Peng, L. Qi, and W. Dou, "BeCome: Blockchain-enabled computation offloading for IoT in mobile edge computing," *IEEE Trans. Ind. Informat.*, vol. 16, no. 6, pp. 4187–4195, Jun. 2019.

[31] X. Xu, C. He, Z. Xu, L. Qi, S. Wan, and M. Z. A. Bhuiyan, "Joint optimization of offloading utility and privacy for edge computing enabled IoT," *IEEE Internet Things J.*, early access, Sep. 26, 2019, doi: 10.1109/JIOT.2019.2944007.

[32] B. Cao, L. Zhang, Y. Li, D. Feng, and W. Cao, "Intelligent offloading in multi-access edge computing: A state-of-the-art review and framework," *IEEE Commun. Mag.*, vol. 57, no. 3, pp. 56–62, Mar. 2019.

[33] M. Yang, W. He, Z. Zhang, Y. Xu, H. Yang, Y. Chen, and X. Xu, "An efficient storage and service method for multi-source merging meteorological big data in cloud environment," *EURASIP J. Wireless Commun. Netw.*, vol. 2019, no. 1, Dec. 2019, Art. no. 241.

[34] X. Wang, T. L. Yang, L. Kuang, X. Liu, Q. Zhang, and M. J. Deen, "A tensor-based big data-driven routing recommendation approach for heterogeneous networks," *IEEE Netw. Mag.*, vol. 33, no. 1, pp. 64–69, Jan. 2019.

[35] X. Xu, R. Mo, F. Dai, W. Lin, S. Wan, and W. Dou, "Dynamic resource provisioning with fault tolerance for data-intensive meteorological workflows in cloud," *IEEE Trans. Ind. Informat.*, early access, Dec. 11, 2019, doi: 10.1109/TII.2019.2959258.

[36] X. Wang, L. T. Yang, H. Li, M. Lin, J. Han, and B. O. Apduhan, "NQA: A nested anti-collision algorithm for RFID systems," *ACM Trans. Embedded Comput. Syst.*, vol. 18, no. 4, pp. 1–21, Jul. 2019, doi: 10.1145/3330139.

[37] L. Qi, Q. He, F. Chen, W. Dou, S. Wan, X. Zhang, and X. Xu, "Finding all you need: Web APIs recommendation in Web of things through keywords search," *IEEE Trans. Comput. Social Syst.*, vol. 6, no. 5, pp. 1063–1072, Oct. 2019.

[38] X. Xu, Q. Liu, X. Zhang, J. Zhang, L. Qi, and W. Dou, "A blockchain-powered crowdsourcing method with privacy preservation in mobile environment," *IEEE Trans. Comput. Social Syst.*, vol. 6, no. 6, pp. 1407–1419, Dec. 2019.

[39] Q.-V. Pham, L. B. Le, S.-H. Chung, and W.-J. Hwang, "Mobile edge computing with wireless backhaul: Joint task offloading and resource allocation," *IEEE Access*, vol. 7, pp. 16444–16459, 2019.

[40] H. Liu, H. Kou, C. Yan, and L. Qi, "Link prediction in paper citation network to construct paper correlation graph," *EURASIP J. Wireless Commun. Netw.*, vol. 2019, no. 1, pp. 1–12, Dec. 2019, doi: 10.1186/s13638-019-1561-7.

[41] W. Gong, L. Qi, and Y. Xu, "Privacy-aware multidimensional mobile service quality prediction and recommendation in distributed fog environment," *Wireless Commun. Mobile Comput.*, vol. 2018, Apr. 2018, Art. no. 3075849.

[42] L. Qi, X. Zhang, W. Dou, C. Hu, C. Yang, and J. Chen, "A two-stage locality-sensitive hashing based approach for privacy-preserving mobile service recommendation in cross-platform edge environment," *Future Gener. Comput. Syst.*, vol. 88, pp. 636–643, Nov. 2018.

[43] L. F. Maimó, A. H. Celdrán, M. G. Pérez, F. J. G. Clemente, and G. M. Pérez, "Dynamic management of a deep learning-based anomaly detection system for 5G networks," *J. Ambient Intell. Hum. Comput.*, vol. 10, no. 8, pp. 3083–3097, Aug. 2019.

[44] M. A. Rahman, M. M. Hasan, A. T. Asyhari, and M. Z. A. Bhuiyan, "A 3D-collaborative wireless network: Towards resilient communication for rescuing flood victims," in *Proc. IEEE 15th Int. Conf. Dependable, Auton. Secure Comput., 15th Int. Conf Pervasive Intell. Comput., 3rd Int. Conf. Big Data Intell. Comput. Cyber Sci. Technol. Congr. (DASC/PiCom/DataCom/CyberSciTech)*, Orlando, FL, USA, Nov. 2017, pp. 385–390.

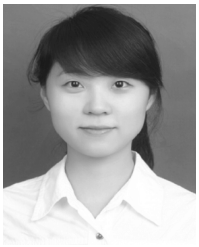
[45] W. Wei, M. A. Rahman, I. F. Kurniawan, A. T. Asyhari, S. M. N. Sadat, and L. Yao, "Immune genetic algorithm optimization and integration of logistics network terminal resources," in *Proc. 3rd IEEE Int. Conf. Robot. Comput. (IRC)*, Naples, Italy, Feb. 2019, pp. 435–436.

MING YANG received the B.S. degree in computer science and technology and the M.S. degree in computer application from the Nanjing University of Information Science and Technology, China, in 2006 and 2011, respectively.

He is currently a Senior Engineer with the Zhejiang Meteorological Information Network Center, Hangzhou, China. His main research interests include high-performance computing, cloud computing, the meteorological IoT, edge computing, machine learning algorithm, and big data processing.

HAO MA received the B.S. degree in atmospheric science and the Ph.D. degree in meteorology from the Ocean University of China, Qingdao, China, in 2006 and 2011, respectively.

From 2011 to 2013, he was an Engineer with the Zhejiang Climate Center, Hangzhou, China, where he became a Senior Engineer, in 2013. His research interests include meteorological big data, cloud computing, climate application, climate prediction, and extended-range weather forecast.



SHUANG WEI received the B.S. degree in computer science and technology, and the M.S. degree in meteorological information technology and security from the Nanjing University of Information Science and Technology, in 2009 and 2012, respectively.

From July 2012 to July 2013, she worked as an Assistant Engineer. She gained the technical title of an engineer, in August 2015. She is currently working with the Zhejiang Meteorological Information Network Center. Her research interests include the analysis and processing of meteorological data, the collaborative quality control and fusion of automatic station and radar data, and the application of data sharing.

YOU ZENG received the B.S. degree in digital media technology from Huazhong Normal University, Wuhan, China, in 2011, and the M.S. degree in design and art from Zhejiang University, Hangzhou, China, in 2014.

She is currently working as an Engineer with the Zhejiang Meteorological Information Network Center, Hangzhou. Her research interests include cloud computing, big data processing, and data visualization.

YEFENG CHEN received the B.S. degree from Hangzhou University, Hangzhou, China, in 1990.

He has been working with the Zhejiang Meteorological Information Network Center, Hangzhou, where he became a Researcher, in 2018. His current research interests are mainly in the areas of meteorological processing, computer networks, information and security system design, and meteorological big data.

YUEMEI HU received the B.S. degree in computer science and technology from Qufu Normal University, Qufu, China, in 2003, and the M.S. degree in computer science and technology from Soochow University, Suzhou, China, in 2006.

She is currently working as a Lecturer with the School of Information Science and Technology, Qufu Normal University. Her research interests include network security, the IoT security, block chain, and service recommendation.