

Multi-objective Optimization of Resource Scheduling in Fog Computing Using an Improved NSGA-II

Yan Sun¹ · Fuhong Lin¹ · Haitao Xu¹

Published online: 3 January 2018

© Springer Science+Business Media, LLC, part of Springer Nature 2018

Abstract In conventional cloud computing technology, cloud resources are provided centrally by large data centers. For the exponential growth of cloud users, some applications, such as health monitoring and emergency response with the requirements of real-time and low-latency, cannot achieve efficient resource support. Therefore, fog computing technology has been proposed, where cloud services can be extended to the edge of the network to decrease the network congestion. In fog computing, the idle resources within many distributed devices can be used for providing services. An effective resource scheduling scheme is important to realize a reasonable management for these heterogeneous resources. Therefore, in this paper, a two-level resource scheduling model is proposed. In addition, we design a resource scheduling scheme among fog nodes in the same fog cluster based on the theory of the improved non-dominated sorting genetic algorithm II (NSGA-II), which considers the diversity of different devices. MATLAB simulation results show that our scheme can reduce the service latency and improve the stability of the task execution effectively.

Keywords Fog computing · Heterogeneous devices · Improved non-dominated sorting genetic algorithm II · Multi-objective optimization · Resource scheduling scheme

1 Introduction

There is an explosive growth in the number of devices getting connected to the network with the rapid development of Internet of Things (IoT). Cisco conservatively estimates that there will be 50 billion devices by 2020 [1]. When these devices simultaneously request

✉ Fuhong Lin
wk2015lfh@163.com

¹ School of Computer and Communication Engineering, University of Science and Technology Beijing, Beijing 100083, China

resource services from the cloud data center, they will take up a large amount of network bandwidth, and data transmission and information access will become slower. Moreover, if some delay-sensitive requests such as medical and emergency are uploaded to the remote cloud for processing, the delay caused by bandwidth constraints and resource bottlenecks of the cloud data center will affect the quality of service (QoS). To overcome this limitation, the concept of fog computing has been proposed [2, 3]. It is a distributed computing model that provides services between traditional cloud data centers and end nodes in the Internet of Things, where cloud services are extended to the edge of the network to reduce the latency and the network congestion [4].

In fog computing, the resource services are provided by geographically distributed devices with certain computing and storage capacities that are located at the edge of the network instead of in a centralized data center as in cloud computing. There are many idle resources in these heterogeneous devices. It is thus very important to make a reasonable resource scheduling to ensure the quality of service and reduce the resource waste. Since the devices that perform the tasks are often heterogeneous and are controlled by different individuals with highly fluctuating behaviors in providing resources [5], the characteristics of the devices performing the tasks should be considered when allocating the resources to each task in fog computing. Therefore, in this paper, we design a resource scheduling scheme specifically for fog computing to realize a more efficient implementation of the tasks. The contributions of this paper are summarized as follows:

1. A new fog computing architecture is proposed, which is divided into the terminal layer, edge layer, and core layer. This architecture can better describe the composition and function of the fog computing.
2. A systematic two-level resource scheduling model is presented, which includes two parts: resource scheduling among various fog clusters and resource scheduling among fog nodes in the same fog cluster.
3. In order to solve multi-objective optimization problems of resource scheduling in the same fog cluster, we propose a novel resource scheduling scheme that efficiently reduces the service latency and improves the overall stability of task execution using an improved non-dominated sorting genetic algorithm II (NSGA-II).

The remainder of this paper is organized as follows: In Sect. 2, the architecture of fog computing is introduced. Then, we present a two-level resource scheduling model in fog computing in Sect. 3. In Sect. 4, an improved NSGA-II is proposed to solve multi-objective optimization problems of resource scheduling in the same fog cluster. In Sect. 5, some simulations show the effects of our proposed scheme on reducing the service latency and improving the stability of task execution. Our conclusions are presented, and future research directions are proposed in Sect. 6.

2 Fog Computing Architecture

Fog computing is located between the end node and the cloud data center. Therefore, the existing architecture of fog computing is typically divided into three layers, which are the cloud, fog, and terminal node (from top to bottom) [6, 7]. To better describe the architecture, we first make some simple but realistic assumptions:

- Fog nodes are local resources, which include network connection devices such as intelligent routers and gateways, and mobile terminal devices such as mobile phones,

tablets and video surveillance cameras. These devices have the abilities of computing, storage, and network connectivity [2]. They can be used not only as resource providers but also as resource consumers.

- Fog clusters consist of local fog nodes. Fog clusters can be interconnected and each of them is linked to the cloud.
- IoT devices, such as sensors, are usually used to obtain information, capture data, and request services, but they do not have the ability to store or calculate.
- The fog nodes can share the network, computational, and storage resources among themselves according to the requirements of the tasks [2].
- A manager node ideally knows all of the information, including the idle resources and the willingness of each device to contribute resources. The manager instructs every fog node to allocate its tasks to other nodes.

In this paper, the architecture of fog computing is divided into the terminal layer, edge layer, and core layer, as shown in Fig. 1.

The terminal layer is mainly composed of resource requestors, including IoT devices such as sensors and traffic lights, and mobile terminal devices such as mobile phones and laptop computers. The latency-sensitive and simple tasks of these devices request resources from the edge layer, whereas the complex tasks request resources from the core layer (cloud data center).

The edge layer is primarily composed of fog resource providers, including gateways, routers, and base stations. Specifically, we find that the edge layer and the terminal layer

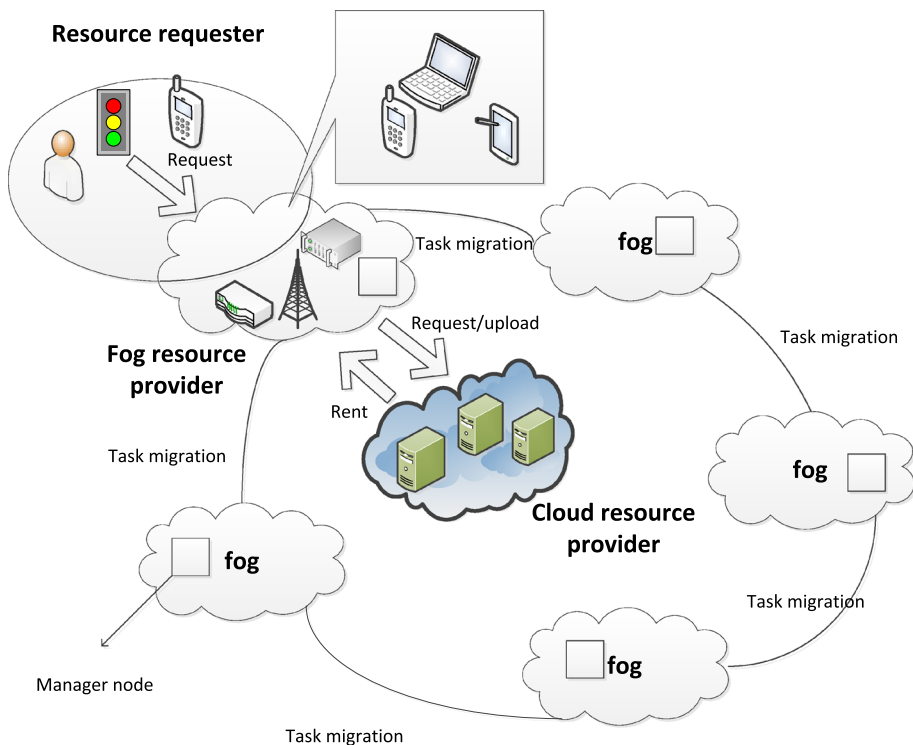


Fig. 1 The architecture of fog computing

partially intersect because the mobile terminal devices in these intersections are not only fog resource requesters but also resource providers. These devices are willing to contribute their own resources to provide external services. The manager at this level is responsible for allocating appropriate resources to process the latency-sensitive tasks from the terminal layer and to provide temporary storage. When the resources of this layer cannot meet the resource requirements of the terminal layer, the edge layer will request resource services from the core layer. In peak times, the capabilities of those fog clusters cannot efficiently serve the resource requester. Therefore, the fog resource provider can extend its capabilities by renting resources from cloud resource providers [8].

The core layer comprises cloud resource providers, including many large servers with many resources and strong task-processing and storage capabilities. They accept resource requests from the edge layer to handle complex computing tasks and large-scale storage tasks. They can outsource computation and storage resources to the edge layer. The emergence of fog computing can greatly ease the pressure on the cloud data center, and as a result, the cloud can focus on handling the permanent and large-scale storage of data and computationally extensive analysis of the tasks from resource requesters. The core layer is now accessed in a periodic and controlled manner, which leads to the efficient and improved utilization of the cloud resources [9].

3 Resource Scheduling Model in Fog Computing

In this paper, resource scheduling in fog computing is divided into two levels: one is the scheduling among fog clusters, which specifies which fog cluster will perform the task when it arrives. The other level of resource scheduling is within the same fog cluster, which specifies which fog resource provider will perform the task when it arrives.

3.1 Resource Scheduling Among Fog Clusters

In the first level of scheduling, as shown in Fig. 1, there is a manager node in each fog cluster, which is a resource scheduler for the tasks submitted from resource requesters at the terminal layer.

The manager is similar to the local resource coordinator in [10], which is selected according to the CPU performance and the battery lifetime. The managers in different fog clusters can interact and communicate with one another and are responsible for the reasonable migration of tasks among the fog clusters. When a user requests resources from the fog clusters, the managers need to communicate with one another and assign the appropriate resources of the fog clusters to perform the tasks of users (i.e., make a reasonable resource allocation), which is called resource scheduling among the fog clusters or the first level of resource scheduling.

Resource scheduling among the fog clusters follows the following two principles: proximity and the resource utilization of fog clusters. The farther the distance between the user who requests resources and the fog cluster is, the longer the required communication delay is. Therefore, the fog cluster that is closer to the resource requester should be chosen to perform the tasks. Therefore, we consider the principle of proximity to reduce the communication overhead between the fog clusters.

Although a short communication distance leads to a reduced communication delay, if the resource utilization of the fog cluster that is closest to the users has reached the upper

threshold, the users may not be able to bear the waiting time. Therefore, when scheduling resources among the fog clusters, we should also consider the resource utilization of fog clusters.

3.2 Resource Scheduling Among Nodes in the Same Fog Cluster

The second level of resource scheduling is within the same fog cluster. The manager of a fog cluster is responsible for managing the fog nodes in the fog cluster. When a service request arrives, it will be divided into several tasks. This paper mainly focuses on the resource scheduling of this level, i.e., which fog node will perform which task.

3.2.1 Problem Description of Resource Scheduling Among the Fog Nodes in the Same Fog Cluster

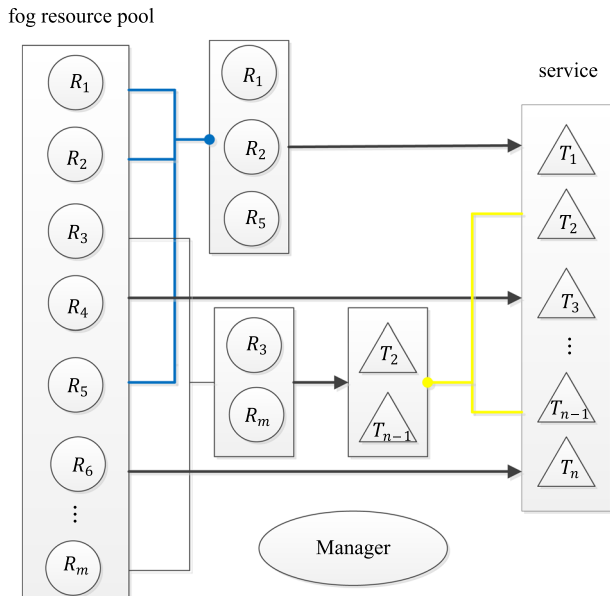
As shown in Fig. 2, the issue of resource scheduling within the same fog cluster is to allocate the appropriate resources for these tasks according to the scheduling goal.

In the fog resource pool, the set of resources is expressed as $R = \{R_1, R_2, \dots, R_j, \dots, R_m\}$. In our paper, one resource represents one fog node in the fog cluster. Resource requesters request resources service by service [10]. If the resource requester r_k submits a service s_k , this service is separated into N related tasks (i.e., sub-tasks), such as computing and data downloading by the manager node, which can be described as $T = \{T_1, T_2, \dots, T_i, \dots, T_n\}$.

Fog computing is a very time-sensitive model [11], and thus, when scheduling resources, we must fully consider the service latency. Moreover, there is great uncertainty in the choice of fog node to perform a task because it is controlled by a person with high subjectivity. Based on the above characteristics, we have designed the following two scheduling goals:

1. Reduce the service latency effectively;

Fig. 2 Resource scheduling diagram



2. Improve the overall stability of the task execution.

3.2.2 Related Mathematical Model

We have established the following mathematical models based on the above two scheduling goals:

Definition 1 Calculation model for service latency

Compared to cloud computing, which can provide high-capacity storage and large-scale computing, fog computing focuses on providing low-latency and real-time services to billions of IoT devices at the edge of the network. The main concern of users is the completion time of the service that they have requested.

Here, we define the completion time of a service as the service latency t_{ser} . The completion time of a service is equal to the sum of the execution time t_{exe} and the transmission time t_{tra} of all of the tasks, which can be expressed as:

$$t_{ser} = t_{exe} + t_{tra}. \tag{1}$$

When we upload tasks to the cloud, because WAN covers a large geographical area from the edge to the core, the communication delay and constrained bandwidth should be taken into account [12]. While the fog device is adjacent to the end user, the LAN communication delay can be ignored. Therefore, the transmission time can be ignored.

In fog computing, many heterogeneous devices cooperate among them and with the network to perform storage and processing tasks. These devices have different calculation capabilities. Furthermore, how fast a fog node executes a task depends on how well the heterogeneous fog node architecture matches the task requirements and features according to [13, 14]. It means the speed of a fog node will be different for different subtasks. Thus, the calculation capability of fog nodes in the fog resource pool is denoted as a two-dimensional matrix, C , in which an element $C(T_i, R_j)$ represents the calculation speed at which fog node R_j executes task T_i . $A(T_i)$ represents the number of instructions to be processed in task T_i . Therefore, the execution time of task T_i running on resource node R_j , denoted as $t(T_i, R_j)$, can be expressed as $A(T_i)/C(T_i, R_j)$. In addition, we consider $l_{i,j}$ represents the condition of task allocation:

$$l_{i,j} = \begin{cases} 1, & \text{if } T_i \text{ is allocated on } R_j \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

For a service composed of N related tasks, the execution time of R_j , denoted as t_j , can be calculated by Eq. (3):

$$t_j = \max_{i=1}^N \left(l_{i,j} * \frac{A(T_i)}{C(T_i, R_j)} \right). \tag{3}$$

Therefore, the service latency t_{ser} should depend on the longest execution time of all resources:

$$t_{ser} = \max_{j=1}^M t_j. \tag{4}$$

Definition 2 Calculation model for stability

In the fog computing environment, the resources originate from different private owners rather than a big data center. The private resource provider may leave the fog cluster or break down arbitrarily and selfishly. To ensure the smooth implementation of tasks, the reputation of fog nodes should be considered. In this paper, reputation is an index that quantifies the degree of a device's reliability and is described using the following equation:

$$rep_j = \begin{cases} \omega_1 p_j^s + \omega_2 p_j^c & N_t < N_{th} \\ p_j^c & N_t \geq N_{th} \end{cases}, \quad (5)$$

subject to $\omega_1 + \omega_2 = 1$, where p_j^s and p_j^c are the reputation evaluation indexes of fog node j . They have different weight values, named ω_1 and ω_2 . N_t is the total number of tasks the resource node R_j has already performed. N_{th} is the threshold value, which can be set as a constant.

p_j^s represents statistical probability that R_j will stay in the fog cluster and perform tasks steadily within the unit time T according to the previous situation statistics of the task execution. It is the statistics average value after the resource node has performed a certain number of tasks. In addition, when the number of tasks the resource node has already performed is too small to reflect the actual reputation ($N_t < N_{th}$), the cognitive probability p_j^c is considered as shown in Eq. (5). p_j^c is defined as the cognitive probability that R_j will stay in the fog cluster and perform tasks steadily within the unit time T according to the type of devices. Heterogeneous devices have different cognitive probability. The mobile devices have lower cognitive probability, because they are in motion and may leave the fog cluster at any time leading to the suspension of the task. Computers, laptops and other fixed devices have higher cognitive probability, because they are located in relatively stable positions and will not easily leave the cluster and stop the task. Laptop can be used in static mode as well as mobile. Here, these devices are divided into three categories: mobile node, static node, semi mobile node. Based on our real experiments in different wired and wireless networks (broadband, WiFi, 3G, and 4G LTE-A), we conclude that the cognitive probability of a static node (desktop computer or server) is approximately 1.5 times the probability of a mobile node (smartphone and similar devices). On the other hand, the cognitive probability of a semi mobile node (tablet and laptop) is approximately 1.25 times the probability of a mobile node.

In the above model for service latency, t_j means the execution time of fog node R_j , which can be calculated as Eq. (3). Therefore, $(rep_j)^{\frac{t_j}{T}}$ should be the probability of successfully completing the task on R_j . For example, we assume that the unit time T is 2 s, and the execution time is 4 s. If $p_j^s = 0.9$, the probability that tasks can be completed smoothly on R_j should be $(0.9)^{\frac{4}{2}} = 0.9 * 0.9$.

In our paper, the stability of the fog node R_j , denoted as sta_j , is defined as the probability that R_j can complete the task smoothly. It can be expressed as follows:

$$sta_j = (rep_j)^{\frac{t_j}{T}}. \quad (6)$$

Here, sta_{ove} represents the overall stability of the task execution. For M resources, sta_{ove} depends on the lowest resource stability, which is described as follows:

$$sta_{ove} = \min_{j=1}^M sta_j. \quad (7)$$

Definition 3 Constraint condition

In order to ensure every task can be completed in required time, one constraint condition need to be met as shown in Eq. (8), where t_i^{req} is the required completion time of T_i :

$$t(T_i, R_j) \leq t_i^{req}. \tag{8}$$

3.2.3 Description of Objective Functions

The optimal resource scheduling scheme in fog computing is a multi-objective optimization problem that minimizes the service latency and maximizes the overall stability as shown in the following equations, (9) and (10):

$$u = \min t_{ser} \tag{9}$$

$$v = \max sta_{ove}. \tag{10}$$

Unlike the single-objective optimization problem which provides only a single optimal solution, the multi-objective optimization problem will provide a set of points known as the Pareto optimal set, which represents the solutions between these objectives.

4 Optimal Scheduling of Resources Using the Improved NSGA-II

To solve multi-objective optimization problems, NSGA-II [15] is a widely adopted algorithm. The basic idea of the NSGA II algorithm is to construct the individual non-dominated sets of the chromosome population, and non-dominated sets in different levels are sorted by the crowding distance of the individuals. The individual with higher non-dominated sets level and larger crowding distance has the higher priority to the next level. The individual of the entire population will suffer the iteration stops or the evolution for several times until meeting the precedent conditions of algorithm. The individual of the population obtained finally is the optimal solution corresponding to the solved problem.

After applying traditional NSGA-II to the problem of resource scheduling based on multi-objective optimization in fog computing, we find that it is not satisfied with the distribution of the solution in the optimal solution set. In order to make the distribution of the individuals in the Pareto optimal frontier more uniform, we have improved the traditional NSGA-II by establishing a new crowding distance formula.

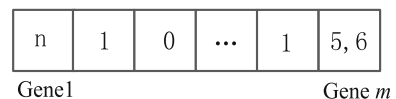
In this paper, the improved NSGA-II algorithm is used to solve the multi-objective optimization problem related to the resource scheduling presented in Sect. 3.2.3.

4.1 Encoding and Initialization

In this paper, every chromosome is taken as a solution of resource-task scheduling in our design.

As shown in Fig. 3, chromosome X_k consists of m genes, and each gene represents one resource. The length of the chromosome is set to be equal to the number of resources, meaning that each chromosome consists of m resources. The value of each gene

Fig. 3 Chromosome X_k



corresponds to the task IDs assigned for this resource. If the resource is not allocated to any task, then the value is set to 0.

This chromosome is encoded as $\{ 'n', '1', '0', \dots, '1', '5', '6' \}$, which can be explained as follows: R_1 is assigned to T_n , R_2 and R_{m-1} are assigned to T_1 , R_3 is not assigned to any task, and R_m is assigned to T_5 and T_6 .

The initial population is randomly generated. If the size of the population is set to SCALE (each generation of the population contains SCALE individuals), it will randomly generate m genomes with the length of SCALE.

4.2 Evaluate Objective Functions

In this paper, we consider two standards to design the fitness function: the service latency and the overall stability of the task execution. The fitness values of the objective functions (9) and (10) are calculated for each individual.

4.3 Construct a Non-dominant Set

Deb et al. [15] proposed the construction method of the non-dominated set, and the main steps to construct the non-dominated set are presented as follows:

- Step 1 Determine the initialization parameters, and the population size SCALE, and set the attribute of the dominated number of individual chromosomes in the tagged population; if $dominated = 0$, then the individual set dominated by the individual chromosome is empty.
- Step 2 Sequentially select an individual chromosome in the population, and compare it to other individuals in the population in terms of the dominance relation. If it is dominated by the compared individual, then for the chromosome attribute, $dominated = dominated + 1$; if it dominates the compared individual, then the compared individual will be added to the individual set dominated by the chromosome.
- Step 3 Repeat step 2 until the dominated attribute of the N chromosomes and the individual set dominated by them have been processed.
- Step 4 Traverse the population and add the chromosome with the dominated attribute of 0 to the rank 1.
- Step 5 Sequentially select the individual chromosome in the rank established in the previous step, and at the same time, the attribute of all of the individuals in the set of individuals is operated by auto-decrement, namely, $dominated = dominated - 1$. If $dominated = 0$, then the individual is added to the rank of next level.
- Step 6 Repeat step 5 until the dominated individual set of the individual chromosome is empty.

4.4 Improved Crowding Distance Calculation

In the fog computing resource scheduling based on multi-objective optimization, different dimension of these two optimization objectives—the service latency and stability of the task make the individuals of the optimal Pareto front generate relatively big gap on these sub-objectives when calculating the crowding distance. While the traditional NSGA-II algorithm calculates the crowding distance of an individual by calculating the sum of

distance difference between the individual and two individuals next to it in each sub-objective and it does not consider the influence brought by the different dimensions of each sub-objective, which has an impact on crowding distance calculation. Based on the traditional crowding distance operator, this paper carries on normalized process on the fitness of the sub-goals where the individual is located, weakening the influence of the crowding distance calculation because of the difference of the dimension, so as to make the distribution of individuals in optimal Pareto frontier better.

To achieve a uniform measurement between the service latency and the stability, we employ the Simple Additive Weighting technique to normalize these two scheduling goals. For the service latency, its decrease will increase the fitness value of the individual, and thus, its normalization formulation is expressed as Eq. (11):

$$f_-(X_k) = f_-(u) = \begin{cases} 1, & \max u - \min u = 0 \\ \frac{\max u - u(X_k)}{\max u - \min u}, & \max u - \min u \neq 0 \end{cases} \tag{11}$$

For the stability, its increase will increase the fitness value the individual, so its normalization formulation is expressed as Eq. (12):

$$g_+(X_k) = g_+(v) = \begin{cases} 1, & \max v - \min v = 0 \\ \frac{v(X_k) - \min v}{\max v - \min v}, & \max v - \min v \neq 0 \end{cases} \tag{12}$$

Thus, when calculating the crowding distance, the fitness of the sub-objective where the individual is located is normalized, which can further weaken the situation in which the individual distribution is not ideal in the optimal Pareto front due to the difference between the sub-objectives.

Therefore, the improved formula of the crowding distance of the individual is calculated as follows:

$$i_d = |f_-^{i+1} - f_-^{i-1}| + |g_+^{i+1} - g_+^{i-1}|. \tag{13}$$

Then, the improved NSGA-II can be applied to the multi-objective optimization of fog computing resource scheduling model mentioned in Sect. 3.2.3. We look for the Pareto optimal front of the multi-objective optimization problem through the improved NSGA-II. Under the corresponding scheduling scheme for all individuals in this frontier, the two optimization objectives, the service latency and the stability of the task could achieve Pareto optimality. The basic flow of the process is shown in Fig. 4.

5 Simulations

In this section, we conduct the experiments and verify the effectiveness of our proposed resource scheduling scheme. To evaluate the performance of our scheme, we compare our scheme with two existing methods, a fog-based IOT resource management model presented by Mohammad Aazam [16], and a random scheduling algorithm. The performance index used for comparison is average latency and average stability.

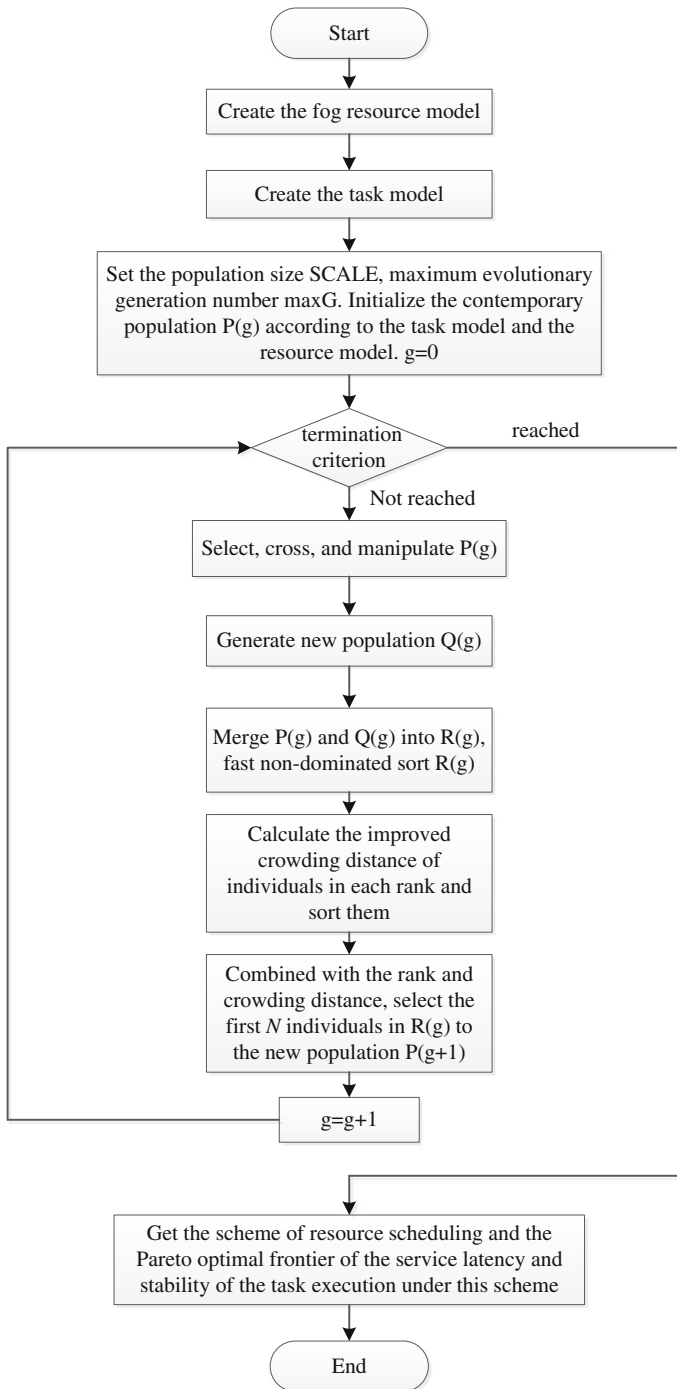


Fig. 4 The basic flowchart of fog resource scheduling based on the improved NSGA-II

5.1 Simulation Scenario

The simulations are performed on a Sun Java SE 7 VM running on Windows 7 PC with an AMD 4.1 GHz CPU and 8 GB memory space.

In the experiments, we consider ‘WordCount’ job as the service request. In our proposed scheme, the calculation speed of resources can be described as the number of instructions that resources can execute per second. The number of instructions per task, tasks, and resources are generated within certain ranges as shown in Table 1. In addition, some related important parameters used for our proposed scheme are described in Table 2.

5.2 Evaluation Methodology

In order to evaluate the performance of our proposed scheme, the following approaches are used:

RSS-IN: It is our scheme for resource scheduling based on the improved NSGA-II.

Random: It selects one solution for resource scheduling randomly.

FIRMM: It is a fog-based IOT resource management model aimed at scheduling and managing resources efficiently and in time.

We assume that population size = 100; crossover probability = 0.85; mutation probability = 0.1. According to [17], the convergent criterion was that the best fitness value would not improve over the last 30 iterations.

5.3 Numerical Results

In order to evaluate the performance of the schemes mentioned above, we use Ganglia system to monitor the status of the fog cluster. The performance has been evaluated in terms of average service latency and average stability. These two evaluation indexes are the results averaged over 100 independent runs. In addition, the Friedman test method is used to assess whether there are significant performance differences between the evaluation result samples obtained by these three schemes [18]. It is a nonparametric statistical inference technique. We apply a tool called SPSS18.0 for providing statistical analysis and calculation. The original hypothesis H_0 we assume is that there are no significant differences between the samples. By calculating the value of the test statistic, we make the decision to reject the original hypothesis when the significance level is 0.05. Therefore, there are significant differences in the performance of three scheduling schemes through the analysis results obtained by Friedman test.

Table 1 Parameters for the resources and tasks

Parameter	Description	Value range
M	The number of resources in the resource pool	[50, 300]
N	The number of tasks	[10, 400]
$A(T_i)$	The number of instructions in task T_i	$[2, 10] \times 10^9$ instr
$C(T_i, R_j)$	Processing rate of T_i on R_j	$[0, 1.5 \times 10^9]$ instr/s

Table 2 Related parameters of our resource scheduling scheme

Parameters	Description	Value
p_j^s	Statistical probability	[0.7, 0.95]
p_j^c	Cognitive probability	[0.6, 0.9]
T	Unit time	2 s
ω_1	The weight value of statistical probability	0.8, 0.5, 0.2
ω_2	The weight value of cognitive probability	0.2, 0.5, 0.8
t_i^{req}	The required completion time of task T_i	[20, 30] s
N_{th}	The threshold value of the number of tasks performed by the fog node	100

The average service latency and the average stability of the task execution under different task numbers are shown in Figs. 5a, b, respectively. As shown in Fig. 5a, when the number of tasks is small, the average service latency of the three schemes is almost the same. However, when the number is large, the advantages of our scheme are obvious. As the number of tasks increases, the average service latency of the proposed scheme is shorter than those of the other two schemes. The average stability of the task execution of the three schemes under different loads is presented in Fig. 5b. As shown in Fig. 5b, the average stability of the task execution of our scheme is always higher than those of Random and FIRMM, because we consider the reputations of the service providers when scheduling resources to the tasks. In addition, the stability decreases as the number of tasks grows, as shown in Fig. 5b, because the large number of tasks increases the risk of task failure.

Figures 6a, b show the average service latency and stability achieved by these three schemes under the increasing number of resources. In Fig. 6a, the average service latency of these three schemes decreases with the increasing number of resources. The stability increases as the number of resources grows, as shown in Fig. 6b. The trend is consistent with those seen in our simulation experiments. The results observed in all of these figures show that the RSS-IN proposed in this paper has better performance than the random scheme and FIRMM in most cases. For RSS-IN, the more resources there are in the system, the more scheduling solutions will be available. Therefore, RSS-IN can select the better solutions (which have low service latency and high stability) to execute the tasks. The service latency will be lower and the overall stability of the task execution will be higher when there are more resource provisions.

Figures 7 shows the performance of our scheme in terms of the average stability when varying the weights of the reputation evaluation indexes. When $\omega_1 > \omega_2$, the reputation evaluation of the fog node is more dependent on the statistical probability p_j^s compared with the cognitive probability p_j^c of the node. As shown in Fig. 7, when the number of tasks assigned to the node does not exceed the threshold N_{th} , the greater the value of ω_2 is, the higher the stability of the task execution will be. The reason is that the evaluation on the reputation of the node through the cognitive probability is more accurate than that through the statistical probability. Statistical probability cannot reflect the true reputation situation of the node due to the lack of statistical samples when the number of tasks assigned to the resource node is below the threshold N_{th} . A node with higher reputation can be selected to perform the task by the way of giving the cognitive probability a higher weight. Therefore, when the number of tasks is less than the threshold, cognitive probability should be given a

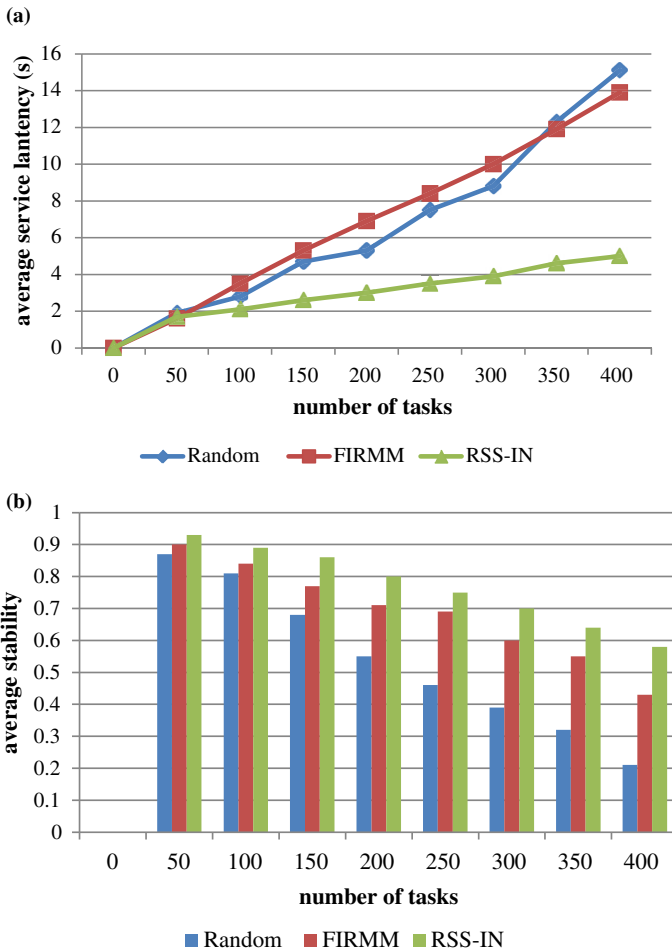


Fig. 5 **a** Average service latency of different task numbers, **b** average stability of different task numbers (the number of resources = 100, $\omega_1 = 0.8$ and $\omega_2 = 0.2$, 100 independent runs)

higher weight to make the evaluation of the node more accurate. In this way, the stability of the task execution can be improved effectively.

6 Conclusion

Fog computing frees up the cloud data center by performing latency-sensitive tasks from the growing end users at the edge of the network. Fog resources need to be used rationally. In this paper, we established a two-level resource scheduling model among the fog clusters and fog nodes. A resource scheduling scheme based on an improved NSGA-II is designed to reduce the service latency and improve the stability of the task execution. In addition to taking into account the quality of service for the resource requesters, the costs of them

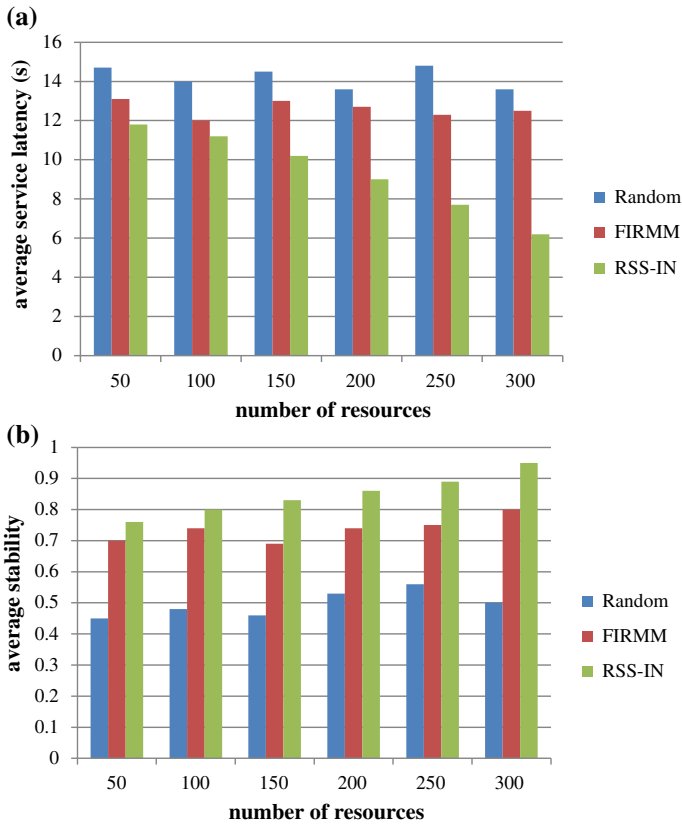


Fig. 6 **a** Average service latency of three schemes under different resource numbers, **b** average latency of three schemes under different resource numbers (the number of tasks = 200, $\omega_1 = 0.8$ and $\omega_2 = 0.2$, 100 independent runs)

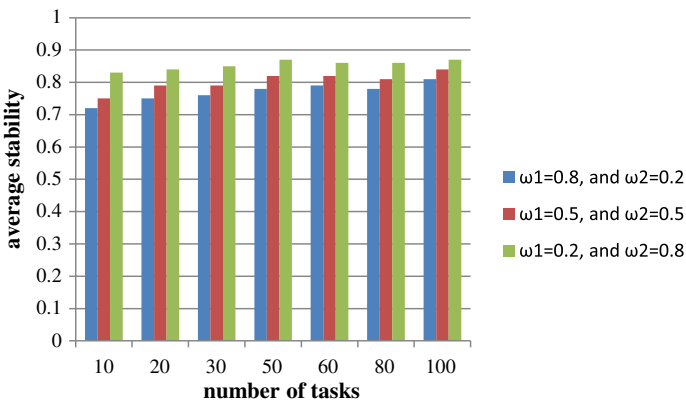


Fig. 7 Average stability with different weights of reputation evaluation indexes (the number of resources = 100, $N_{th} = 100$, 100 independent runs)

should also be considered. Therefore, in the future, we will further improve our proposed resource scheduling scheme to reduce the cost of resource requesters.

Acknowledgements We thank American Journal Experts (AJE) for its linguistic assistance during the preparation of this manuscript.

References

1. Evans, D. (2011). The internet of things: How the next evolution of the internet is changing everything[J]. *CISCO White Paper, 2011*(1), 1–11.
2. Bonomi, F. (2011). Connected vehicles, the internet of things, and fog computing[C]. In *The eighth ACM international workshop on vehicular inter-networking (VANET)*, Las Vegas, USA (pp. 13–15).
3. Vaquero, L. M., & Rodero-Merino, L. (2014). Finding your way in the fog: Towards a comprehensive definition of fog computing[J]. *ACM SIGCOMM Computer Communication Review, 44*(5), 27–32.
4. Gupta, H., Dastjerdi, A. V., Ghosh, S. K., et al. (2016). iFogSim: a toolkit for modeling and simulation of resource management techniques in internet of things, edge and fog computing environments[J]. arXiv preprint [arXiv:1606.02007](https://arxiv.org/abs/1606.02007).
5. Aazam, M., & Huh, E. N. (2015). Dynamic resource provisioning through fog micro datacenter[C]. In *2015 IEEE international conference on pervasive computing and communication workshops (PerCom workshops)* (pp. 105–110). IEEE.
6. Stojmenovic, I., & Wen, S. (2014). The fog computing paradigm: Scenarios and security issues[C]. In *2014 federated conference on computer science and information systems (FedCSIS)* (pp. 1–8). IEEE.
7. Hu, P., Ning, H., Qiu, T., et al. (2017). Fog computing-based face identification and resolution scheme in internet of things[J]. *IEEE Transactions on Industrial Informatics, 13*(4), 1910–1920.
8. Pham, X. Q., & Huh, E. N. (2016). Towards task scheduling in a cloud-fog computing system[C]. In *2016 18th Asia-Pacific Network Operations and Management Symposium (APNOMS)* (pp. 1–4). IEEE.
9. Sarkar, S., & Misra, S. (2016). Theoretical modelling of fog computing: A green computing paradigm to support IoT applications[J]. *IET Networks, 5*(2), 23–29.
10. Nishio, T., Shinkuma, R., Takahashi, T., et al. (2013). Service-oriented heterogeneous resource sharing for optimizing service latency in mobile cloud[C]. In *Proceedings of the first international workshop on Mobile cloud computing & networking* (pp. 19–26). ACM.
11. Al Faruque, M. A., & Vatanparvar, K. (2016). Energy management-as-a-service over fog computing platform[J]. *IEEE Internet of Things Journal, 3*(2), 161–169.
12. Deng, R., Lu, R., Lai, C., et al. (2015). Towards power consumption-delay tradeoff by workload allocation in cloud-fog computing[C]. In *2015 IEEE International Conference on Communications (ICC)* (pp. 3909–3914). IEEE.
13. Xu, Y., Li, K., He, L., et al. (2013). A DAG scheduling scheme on heterogeneous computing systems using double molecular structure-based chemical reaction optimization[J]. *Journal of Parallel and Distributed Computing, 73*(9), 1306–1322.
14. Xu, Y., Li, K., Khac, T. T., et al. (2012). A multiple priority queueing genetic algorithm for task scheduling on heterogeneous computing systems[C]. In *2012 IEEE 14th International Conference on High Performance Computing and Communication & 2012 IEEE 9th International Conference on Embedded Software and Systems (HPCC-ICES)* (pp. 639–646). IEEE.
15. Deb, K., Pratap, A., Agarwal, S., et al. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II[J]. *IEEE Transactions on Evolutionary Computation, 6*(2), 182–197.
16. Aazam, M., & Huh, E. N. (2015). Fog computing micro datacenter based dynamic resource estimation and pricing model for IoT[C]. In *2015 IEEE 29th international conference on advanced information networking and applications (AINA)* (pp. 687–694). IEEE.
17. Wang, D., Yang, Y., & Mi, Z. (2015). A genetic-based approach to web service composition in geo-distributed cloud environment[J]. *Computers & Electrical Engineering, 43*, 129–141.
18. Friedman, M. (1937). The use of ranks to avoid the assumption of normality implicit in the analysis of variance[J]. *Journal of the American Statistical Association, 32*(200), 675–701.



Yan Sun is pursuing his Ph.D. degree in Department of Communication Engineering, School of Computer and Communication Engineering (SCCE), University of Science and Technology Beijing (USTB). His research interests include resource allocation of cloud computing, game theory, network redundancy elimination.



Fuhong Lin received his M.S. and Ph.D. degrees from Beijing Jiaotong University, China in 2006 and 2010, respectively, both in electronics engineering. He is currently an associate professor in the Department of Communication Engineering, University of Science and Technology Beijing, China. His research interests include wisdom network, social network and P2P network.



Haitao Xu received his Ph.D. degree in the Department of Communication Engineering, School of Computer and Communication Engineering (SCCE), University of Science and Technology Beijing. He is now an associate professor in the Department of Communication Engineering, University of Science and Technology Beijing, China. His research interests include wireless network, game theory and big data.