EAI Endorsed Transactions
on Internet of Things

Research Article EAI.EU

Service Placement Optimization Based on Evolutionary Algorithm in Fog Computing

Jiamin Niu[1], Gang Liu[1,*], Lin Yu[1] and Jiawei Wang[1]

[1]Computer Science and Technology, XIDIAN University, Xi'an, China

Abstract

As an emerging distributed computing paradigm, fog computing provides low-latency and real-time interactive services to end-user or Internet of Things(IoT) devices at the edge of the network. One of the main challenges of fog computing is to select the right fog node to deploy and run IoT application services, which is commonly referred to as the fog service placement problem (FSPP). However most schemes model FSPP as a single objective optimization problem. These single-objective optimization schemes usually cannot meet the needs of increasingly complex engineering practice. In this study, we model the fog service placement problem as a constrained multi-objective optimization problem, which aims to improve the resource utilization of the system and reduce network latency and service placement costs. Secondly, the elitist nondominated sorting genetic algorithm II (NSGA-II) is used to optimize the constrained multi-objective service placement problem. Experimental results show that the proposed scheme is superior to the existing schemes in terms of overall performance.

Keywords: Fog computing, Fog service placement, Multi-objective optimization, NSGA-II.

Received on 02 January 2022, accepted on 08 February 2022, published on 22 February 2022

doi: 10.4108/eai.22-2-2022.173492

*Corresponding author. Email: gliu@xidian.edu.cn

1. Introduction

With the rapid development of the IoT, traditional cloud computing can no longer meet the needs of applications for low latency, real-time interaction and mobility awareness. For this reason, some scholars have introduced a more efficient distributed computing paradigm, which can avoid network bottlenecks, overcome traffic loads and reduce data transmission latency, usually called Fog Computing (FC) [1]. Due to the heterogeneity, resource limitation, and wide geographical distribution of fog devices, how to efficiently organize and manage the resources in fog devices has become a hot research area in the field of fog computing.

One of the challenges of fog computing is to determine which services each fog node should host to meet user requirement , commonly referred to as the Fog Service Placement Problem (FSPP) [2]. It is necessary to consider the heterogeneity of fog nodes, the special attributes of applications, and the diversity that users expect in FSPP. For example, with the increasing complexity of scientific research and engineering practice, it is required to consider a set of optimal balanced solutions among several mutually exclusive objectives to meet the needs of practical applications when formulating of fog service placement strategy. These aspects complicate the placement problem, and also make the solution of the placement problem a challenging problem.

Due to these challenges, existing researches on fog service placement usually have limitations in terms of practicability and performance. These researches mainly optimize service placement in terms of the cost, resource utilization, latency, QoS and power consumption [3][4][5][6]. Most of them model FSPP as a single objective optimization problem, only focusing on optimizing a performance indicator of the fog service placement. Although a few researches have considered multi-objective optimization of FSPP, they usually use linear summation function to convert multiple objective functions into single objective optimization problems. Other multi-objective evolutionary algorithms with higher efficiency and better performance have not attracted extensive attention from researchers, such as NSGA-II. Salaht et al. [20] also pointed out that the modeling and optimization of the multi-objective fog service placement problem is one of the important research directions in the field of fog computing.

This paper models the service placement problem in a multi-objective optimization in the fog computing environment, and proposes a fog service placement optimization algorithm based on NSGA-II to solve this problem. The main contributions are as follows:

(1) We build a multi-objective mathematical model of the fog service placement problem, set up three optimization objectives of the system: resource utilization, service placement cost and network latency, and take the total amount of resources required by all services placed on the fog equipment must be less than or equal to the resources owned by the equipment as the constraint condition.

(2) This paper proposes an optimization algorithm for fog service placement based on NSGA-II. The experimental results show that the performance of the proposed solution is greatly improved compared with the existing weighted sum genetic algorithm.

The remainder of this paper is organized as follows. Section 2 reviews the related work on fog computing. System model and problem modeling are presented in Section 3. The fog service placement optimization algorithm based on NSGA-II is described in Section 4. Section 5 discusses our simulation results and result analysis. Finally, Section 6 summarizes the whole research work.

2. Related work

Regarding the problem of service placement in the fog computing environment, most of the existing optimized service placement algorithms focus on minimizing the delay perceived by users, reducing the cost of service placement, or improving the resource utilization of the fog layer. Whether the quality of service can be improved as the evaluation standard of the algorithm. In order to optimize the fog service placement problem, the existing research uses a variety of optimization algorithms, such as linear programming [3][7][8], complex network theory [9], Markov process[10], genetic algorithm [11][12] and Petri net [13], etc.

To our knowledge, only a few researches have focused on the modeling and optimization of multi-objective FSPP. Skarlat et al. [12] introduced the concept of fog community, and proposed the optimal service placement scheme between communities. This scheme takes response time and service quality as the optimization objectives, and uses a genetic algorithm to determine whether to place the service in the community or to transmit the service to the adjacent community for each community. Yang et al. [15] constructed a mathematical model of cost-aware service placement problem, and the problem was solved based on genetic algorithm, integer linear programming and heuristic greedy algorithm. Although the above-mentioned solutions model a multi-objective FSPP, they use a weighted sum genetic algorithm to optimize the proposed model, which is a single-objective optimization problem essentially. The weighted-sum genetic algorithm performs poorly in solving FSPP, as shown in the following experiments.

In order to explore other heuristic algorithms with higher solving efficiency, Moallemi et al. [11] proposed a meta-heuristic algorithm based on NSGA-II to solve the service placement problem. The proposed scheme not only maximizes the service range of each edge node, but also minimizes the waste of resources by reducing the overlap area between these nodes. However, the proposed scheme does not take network latency as the optimization objective. Zhang et al. [16] defined the optimal controller placement problem as a multi-objective optimization problem, optimizing network reliability, load balance among controllers, and low delay between controllers and switches. The adaptive bacterial foraging optimization algorithm is proposed to optimize the problem, but the resource utilization and service placement cost of the fog layer are not considered in the proposed scheme. The FSPP problem was represented as a multi-objective optimization problem by Natesha et al. [17], who applied a genetic algorithm based on an elite approach to solve it. They optimized the energy consumption, service time, and cost, but the proposed scheme failed to account for fog layer resource usage.

3. System model

This section first describes two models related to the FSPP problem: the IoT application model based on microservice and the system model of fog computing. The mathematical model of the multi-objective fog service placement problem is built on the foundation of these two models. Based on these two models, the mathematical models of optimization objectives and constraints of FSPP are proposed.

3.1. An Application Model of IoT based on Micro-service

Some existing researches have processed the placement of virtual machines(VM), virtual data centers or services. The placement of services is different from the other two, which represents a more fine-grained level of the same problem [26], so it is well suited to fog computing environments with limited resources. We use an IoT application model based on microservice, which is composed of a set of small stateless services. These small stateless services communicate with others via messages, and they transmit messages to collaborate on complex tasks [18].

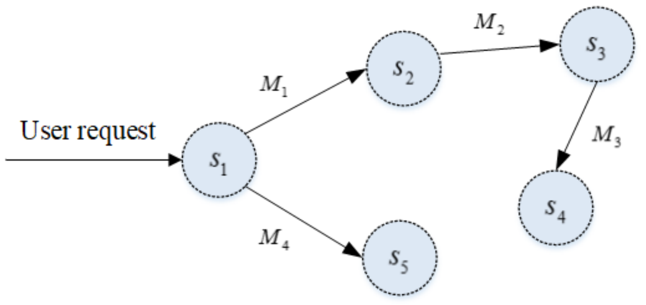

Figure 1. An application model based on micro-services

Each application in the system is modeled as a directed acyclic graph $DG = (S, M)$. The set of points in DG is represented by $S = \{s_1, s_2, \ldots, s_k\}$, denoting that the application consists of k small stateless services, and s_k is the k-th service of the application. A service will be executed only when it receives the request message from the forward service. Following the execution of the service, the application will generate 0 or more new request messages to request new services until no new request messages are generated. According to the various service request sources in reference [19], there are two types of service: 1) entrance service, which is usually initiated by end users or IoT devices; 2) internal service, only requested by other services. In Figure 1, s_1 is an entrance service, while other services are internal services. This article assumes that each application has only one entrance service.

Edge M in DG represents the message passing relationship between services, which is denoted by M_{s_o, s_t}, where s_o is the source service and s_t is the target service. The end user, in particular, is the source service of the entrance service. In this paper, $M_{\emptyset, s_t}$ is used to represent the request message between the user and the entrance service. The request message M_{s_o, s_t} has two important characteristic parameters: $MS_{M_{s_o,s_t}}$ and $MI_{M_{s_o,s_t}}$ represent the size of the message requesting the target service and the workload that the target service needs to execute, respectively. The former determines the transmission time of the request message in the network, while the latter is defined as the number of instructions to be executed [19]. In this paper, we use binary variables $x_{o,t}$ to describe the consumption relationship between the source service s_o and the target service s_t. If the source service consumes the target service, it is 1. Otherwise it is 0. As a result, the consumption relationship between k small stateless services in an application APP_a can be represented by an adjacency matrix A of dimension $k \times k$, which is expressed as follows

$$A = \begin{pmatrix} x_{1,1} & \cdots & x_{1,k} \\ \vdots & \ddots & \vdots \\ x_{k,1} & \cdots & x_{k,k} \end{pmatrix} \quad (3\text{-}1)$$

Furthermore, we use C_{s_k} to denote the quantity of resources required to instantiate the service s_k.

3.2. Fog computing model

In Figure 2, we consider a hierarchical edge infrastructure structure, which usually contains three layers: cloud layer, fog layer, and end end-user layer. Like the traditional cloud data center, the cloud layer has powerful computing and storage functions and provides various IoT application services. The end-user layer is composed of a series of IoT devices (sensors and actuators) or users. These devices or users make requests and get responses to IoT applications. The terminal device or user in this article is static, and its mobility is ignored. The fog layer is composed of network devices between the cloud layer and the end-user layer. These devices have functions such as computing, storage, and communication and can execute service instances.

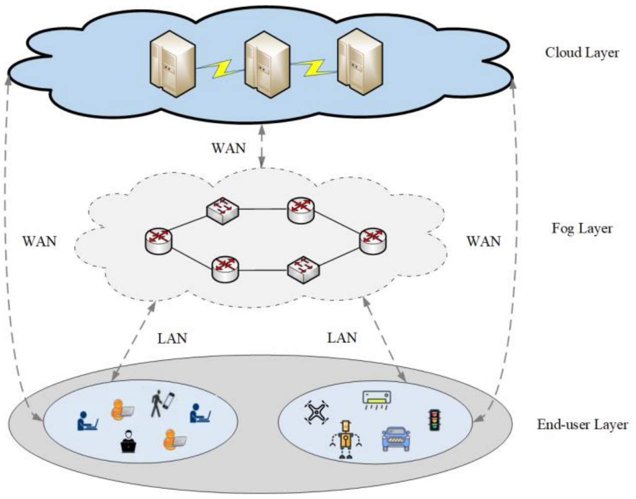

Figure 2. Fog infrastructure structure

In this paper, the fog infrastructure is modeled as an undirected graph $G = (D, E)$. The set of points in G is represented by $D = \{d_1, d_2, \ldots, d_n\}$. There are n fog devices, and d_i represents the i-th fog device in the set. $E = \{e_{i,j}\}$ is the set of edges in the undirected graph G, and $e_{i,j}$ represents the network communication links between the two directly connected fog devices d_i and d_j. We use R_{d_i} to represent the number of resources owned by the device d_i. R_{d_i} is usually a vector, which contains some physical properties of the fog device (for example, the number of cores for the CPU, memory size). In order to reduce the complexity of the model, this paper uses a scalar value to represent the number of resources owned by the device. Another important characteristic parameter of the fog equipment is the processing speed IPT_{d_i}, measured by the number of millions of instructions executed per unit of time. The network communication links is denoted by $e_{i,j}$, and each communication link has two important characteristic parameters $PR_{e_{i,j}}$ and $BW_{e_{i,j}}$, which respectively represent the propagation latency and the network bandwidth of $e_{i,j}$. Therefore, the network delay $L_{e_{i,j}}(size)$ for data transmission between two directly connected fog devices d_i and d_j is calculated as following formula [9]:

$$L_{e_{i,j}}(\text{size}) = PR_{e_{i,j}} + \frac{\text{size}}{BW_{e_{i,j}}} \tag{3-2}$$

where size represents the size of the data packet to be transmitted between the two devices.

The data transmission network latency D_{d_i,d_j} between any two fog devices d_i and d_j in the fog computing network is the sum of the network delays for data transmission in the shortest path between the two fog devices. Dijkstra's Algorithm found the shortest path between two fog devices with the shortest data transmission latency. Its mathematical expression is as follows:

$$D_{d_i,d_j}(\text{size}) = \sum_{\forall e_{g,h} \in \text{shortestPath}(d_i,d_j)} L_{e_{g,h}}(\text{size}) \tag{3-3}$$

We use the binary placement decision variable $p_{i,k}$ to define whether the service s_k is placed on the fog device d_i. The formula is as follows:

$$p_{i,k} = \begin{cases} 1 & \text{If service S is assigned to fog node d} \\ 0 & \text{Other situations} \end{cases} \tag{3-4}$$

We suppose that the system contains t different IoT applications with m various services. The placement of service instances on the fog device is represented by a two-dimensional matrix P of size $n \times m$, where n is the number of fog equipment in the system and m is the number of various services. The service placement matrix P is as follows:

$$P = \begin{pmatrix} p_{1,1} & p_{1,2} & \cdots & p_{1,m} \\ p_{2,1} & p_{2,2} & \cdots & p_{2,m} \\ p_{3,1} & p_{3,2} & \cdots & p_{3,m} \\ \vdots & \vdots & \vdots & \vdots \\ p_{n,1} & p_{n,2} & \cdots & p_{n,m} \end{pmatrix}, p_{i,k} \in [0,1] \tag{3-5}$$

3.3. Optimization model

We optimized the system from three different perspectives: 1) minimize network latency, 2) maximize the system resource utilization, and 3) minimize the service placement costs. This section will describe in detail the mathematical model of the optimization objective we selected, and the mathematical model of the constraints of the FSPP.

Network latency

Fog computing's ability to lower service response time is one of the key reasons for its widespread adoption in numerous areas. Therefore, the first optimization objective is to minimize the network latency Latency, defined as the average value L_{APP_a} of the average response time of each application APP_a in the system. The calculation formula is as follows:

$$\text{Latency} = \frac{\sum_{a=1}^{t} L_{APP_a}}{t} \tag{3-6}$$

where t is the number of IoT applications in the system, L_{APP_A} represents the average response time of the application APP_a. The calculation formula is as (3-7).

$$L_{APP_a} = \frac{\sum_{u=1}^{|num_a^{user}|} F(user_u, APP_a)}{|num_a^{user}|} \tag{3-7}$$

where $|num_a^{user}|$ represents the number of users requesting the application, $F(user_u, APP_a)$ represents the response time of the application APP_a to the user $user_u$, Its value is the interval between the sending time of the user's request to the APP_a and the completion time of the last service in the APP_a. As shown in the application model in section 3.2, the response time of APP_a to the user $user_u$ is determined by communication latency among services contained in the application and calculation delay of the service instance on the fog device. Calculate the response time of the service using the APP_a's entry service as an example. When a user requests APP_a, the gateway d_i^{gw} sends the request to the fog device d_i nearest to the user, which is deployed with s_k for processing. The shortest distance between two fog devices is calculated by Dijkstra's algorithm, meaning that the user requests the closest service. The network latency of the user request message can be expressed as the network latency $D_{d_i^{gw},d_i}(MS_{M_{\emptyset,s_t}})$ required to transmit the request message $M_{\emptyset,s_t}$ between d_i^{gw} and d_i, and its value can be calculated by formula (3-3). The execution time $ET(s_k, d_i)$ of service s_k on fog device d_i is calculated as follows:

$$ET(s_k, d_i) = \frac{MI_{M_{\emptyset,s_t}}}{IPT_{d_i}} \tag{3-8}$$

Therefore, the response time of entry service s_k is $RS(s_k) = D_{d_i^{gw},d_i}(MS_{M_{\emptyset,s_t}}) + ET(s_k, d_i)$. After the execution of s_k is completed, the matrix A can be queried to find the service set consumed by s_k, and then for each service s_k^* in the set, the Dijkstra’s algorithm is used to find the fog device that is nearest to the fog device d_i and the service s_k^* is deployed. Then s_k sends a new request message M_{s_k,s_k^*} to the service s_k^*, and the response time of s_k^* is calculated in the same way as s_k. Repeat the the process until a service no longer generates a request message or the last service of the APP_a completes a response.

Resource utilization

An essential issue in fog computing is to optimize resource utilization while maximizing the number of services placed on fog equipment [20]. The second optimization objective is to maximize resource utilization. It is defined as the ratio of the total resources required by all services deployed in the system to the total amount of resources provided by all fog devices. The mathematical expression is as follows:

$$\text{Resource Usage} = \frac{\sum_{i=1}^{n} \sum_{k=1}^{m} p_{i,k} \times C_{s_k}}{\sum_{\forall d_i \in D} R_{d_i}} \tag{3-9}$$

We transform the problem max(ResourceUsage) into a minimization problem. The value of the resource utilization of the objective function is less than or equal to 1, so we convert maximizing the resource utilization of the system into minimizing the idle resource ratio of the system via formula (3-10). The idle resource ratio has the following definition:

$$\text{Free Resource} = 1 - \text{Resource Usage} \qquad (3\text{-}10)$$

Service cost

Cost-related factors are becoming increasingly important in fog resource management, whether from the perspective of service providers or end users. Therefore, the third optimization objective is to minimize the service cost Cost. The total cost Cost of the service mainly depends on the storage cost β per unit time of the fog device, the number of service instances deployed in the fog layer, and the time τ of the service placement. Cloud data center and fog equipment have the same β [21]. The proposed service placement strategy will solve FSPP periodically, and the life cycle τ of service instance on fog device will end with the deployment of new service placement scheme. Therefore, in this article, τ is the cycle that the service placement strategy solve the fog service placement problem. In summary, the calculation method of service cost is as follows:

$$\text{Cost} = \beta\tau\left(\sum_{k=1}^{m} C_{s_k} + \sum_{i=1}^{n}\sum_{k=1}^{m} p_{i,k} \times C_{s_k}\right) \qquad (3\text{-}11)$$

Constraint condition

The constraint condition of FSPP is that the resources consumed by the services in the fog device must be less than or equal to the resources owned by these devices. The mathematical expression is as follows:

$$\left(\sum_{k=1}^{m} p_{i,k} \times C_{s_k}\right) \leq R_{d_i}, \forall d_i \in D \qquad (3\text{-}12)$$

In summary, our goal is $\min(\text{Free Resource}) \wedge \min(\text{Cost}) \wedge \min(\text{Latency})$. we solve the optimal service placement matrix P under the constraints of formula (3-12). Therefore, the fog service placement problem in this paper can be described as follows:

$$\begin{gathered}\text{P: } \min(\text{Free Resource}, \text{Cost}, \text{Latency}) \\ \text{Subject to } (3-12)\end{gathered} \qquad (3\text{-}13)$$

4. Fog service placement optimization algorithm based on NSGA-II

Because the service placement problem in the fog computing environment is NP-hard [27], heuristic algorithms can study a larger search space and give feasible solutions in polynomial time. Therefore, we use NSGA-II proposed by Deb et al. [22] to solve the FSPP. We will describe the implementation details of fog service placement algorithm based on NSGA-II in this section.

4.1. Individual representation and population initialization

In the implementation of the algorithm, the service placement matrix P of size $n \times m$ is regarded as a chromosome, as shown in formula (3-5). The initial population consists of N randomly generated chromosomes.

4.2. Genetic operator

Crossover operator

The single-point crossover method is used between the two parents for crossover operation. The crossover probability P_c is one of the parameters the crossover process. When a crossover operation is performed, a random value will be generated in the range of [0,1]. Only when the value is less than or equal to the crossover probability P_c will the crossover operation be performed. When performing the crossover operation, the algorithm randomly selects a crossover point r in the matrix defined by formula (3-5), where r is between 1 and n . The crossover point corresponds to the starting number of the row to be exchanged in the matrix, and new offspring are generated by exchanging the r-th row and subsequent rows of the two service placement schemes. Figure 3 shows the structure diagram of the chromosome structure of the two offspring P_i^* and P_j^* produced by the two fathers P_i and P_j after performing the crossover operation, where r represents the crossover point, and $P_{n,m}^i$ is the service placement decision variable of solution P_i, indicating whether the fog device d instantiates service s_m in solution P_i. The meaning of $P_{n,m}^j$ is similar.

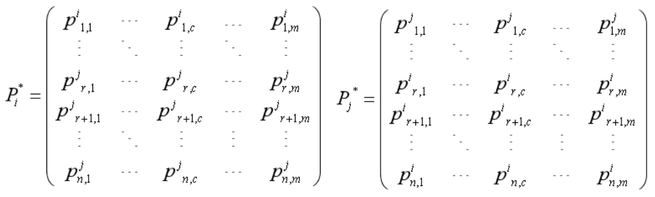

Figure 3. The offspring chromosomes after the crossover operation

Mutation operator

In this paper, the mutation operators in genetic operations can expand the search space and avoid falling into local optima. The mutation operation consists of two parts: randomly increasing or decreasing the number of each service instance by iterating each row of the individual's chromosome that needs to be mutated; randomly selecting a subset of the service in the system and instantiating it in all fog devices.

Modified operator

The individuals in the initial population and the individuals obtained after mutation may violate the constraint (3-12);

that is, the total number of resources required for the services placed on the fog device exceeds device capacity. We define a modified operator to modify it. This operator is an iterative process that randomly deletes deployed service instances from fog devices that violate constraints until the resources consumed by the placed services are less than or equal to the device capacity.

4.3. Fog service placement optimization algorithm based on NSGA-II

The pseudo code of the fog service placement optimization algorithm based on NSGA-II is given in algorithm 1. The input parameter of this algorithm is used to generate the initial population. Since there may be illegal solutions in the randomly generated initial population , it is necessary to correct the illegal solutions in the population using a modified operator (line 1).

Individual fitness is expressed as a vector $\mathrm{fitness} = [f_1(x), \ldots, f_m(x)]^T$in Algorithm 1, which is composed of the value of the individual's objective function. Sorting individuals in the population according to their fitness is the core of the NSGA-II algorithm. In order to screen out the elite individuals to construct the next-generation population, Algorithm 1 introduces two fundamental concepts in the NSGA-II algorithm: fast non-dominated sorting operator and crowded distance.

The fast non-dominated sorting operator [22] divides the population into multiple non-dominated levels and assigns each individual a non-dominated order value(lines 2 and 14). The basic steps of the fast non dominated sorting operator are as follows:

(1) Let $\partial = 1$, then calculate the Pareto optimal set of the population, and set the non-dominated order value of each individual in the Pareto optimal set to ∂, where $\partial = \partial + 1$;

(2) Remove individuals from the population that have been assigned a non-dominated order value, calculate the Pareto optimal set in the current population, and set the non-dominated order value to ∂ for each entity in the set, where $\partial = \partial + 1$;

(3) Repeat step 2 until the population is empty.

After executing the fast non-dominated sorting operator, each individual in the population has a non-dominated order value. The lower a non-dominant order value of an individual, the more adaptable it is to the environment, and the more probable it is to be selected to join the next-generation population.

```
Algorithm 1: Fog service placement optimization algorithm based on NSGA-II
  input  : Population size $populationSize$
           Number of iterations $Number$
  output: Pareto Optimal Solution of Fog Service Placement Problem
1  Randomly generate an initial population $P_t$ whose size is
     $populationSize$, and use a modified operator to modified the illegal
     solution in P
2  Non-dominated sorting of population $P_t$
3  for $i = 1$ to $Number$ do
4      Initialize the set of offspring populations $P_{off} = \emptyset$;
5      for $j = 1$ to $populationSize$ do
6          Use binary tournament to choose two fathers $father1$ and
             $father2$ from $P_t$;
7          Use the crossover operator to perform crossover operations on
             $father1$ and $father2$ to produce two offspring $child1$ and
             $child2$;
8          if $random() < mutation\ probability\ P_{mu}$ then
9              Use mutation operator to perform mutation operation on two
                 children $child1$ and $child2$;
10         end
11         Add the two newly generated offspring to the offspring
             population, $P_{off} = P_{off} \cup \{child1, child2\}$
12     end
13     Combined father and son populations, $P_{off} = P_{off} \cup P_t$;
14     Perform non-dominated sorting on $P_{off}$, calculate the crowding
         distance of individuals in $P_{off}$, and use the non-dominated order
         value and crowding distance of the individuals to sort $P_{off}$;
15     Update the iterative population $P_t$, and take the first
         $populationSize$ individuals with the highest fitness in $P_{off}$ to form
         the next generation parent population
16 end
17 return Individuals with a non-dominant ordinal value of 1 in $P_t$
```

Algorithm 1 calculates the individual crowding distance on line 14 to sort the individuals in the same non-dominated level. Crowding distance, which represents the density of individuals at a non-dominated level, is defined as the Euclidean distance between the current individual and two adjacent individuals. In order to maintain the population diversity, the individuals with large crowding distances are preferred to enter the next generation population when the non-dominated order values of individuals are equal.

After the operation of the non-dominated sorting operator and the crowding distance operator, each individual has two attributes: the non-dominated order value $\mathrm{rank(p)}$ and the crowded distance $\mathrm{distance(p)}$. The individuals in the population are sorted according to the following rules: when either $\mathrm{rank}(p_1) < \mathrm{rank}(p_2)$ or $(\mathrm{rank}(p_1) = \mathrm{rank}(p_2)$ and $\mathrm{distance}(p_1) > \mathrm{distance}(p_2))$ is established for any two individuals p_1 and p_2 in the population, the individual p_1 is better than p_2, that is, the fitness of individual p_1 is higher than p_2.

For the next generation parent population, algorithm 1 uses an elite selection strategy. That is, a new population is created by merging the generated offspring P_{off} with the previous parent population P_t, and the new population is sorted according to the individual's non-dominated order value and crowding distance (line 14). Then the first populationSize individuals are selected from the ordered population to establish the parent population of the next generation (line 15).

5. Performance analysis

5.1. Simulation settings

(1) Network topology: The paper uses the randomly generated Albert-Barabasi network as the network infrastructure in the experiment. This network model is widely used in the research of fog resource management [23]. In the study, the concept of betweenness centrality is introduced to distinguish the devices in the network, and the node with the highest betweenness centrality is selected as the cloud data center. On the contrary, the gateway device is selected from the nodes with low betweenness centrality. The network scale in the experiment is set to 100, and 25 fog devices with low betweenness centrality are selected as the gateway. Users must use the gateway to send requests to IoT applications. Users connected to each gateway follow a uniform distribution.

(2) Network and device parameters: In the experiment, we assume that the propagation delay among fog devices is $U(1,2)$ms, and that propagation delay between fog devices and cloud data centers is $U(15,30)$ms [24]. The bandwidth size in the network link is set to 7500bytes/ms [9]. The resource capacity R_{d_i} of each fog device is $U(4,10)$MB, and the processing speed IPT_{d_i} of the fog device is $U(100,1000)$MIPS [9]. In particular, the resource capacity of d^{cloud} is infinite, and its processing speed is 10000 MIPS. The service instance in the cloud data center and fog equipment has a storage cost per unit time β of 0.004Mb/s, and the optimization cycle τ of FSPP is 2min [24].

(3) IoT application parameters: Each IoT application is composed of $U(2,10)$ small stateless services, each of which requires $U(1,5)$MB of resources to place. The number of instructions to be executed by each service is $U(20000,60000)$, and the size of messages to be transferred between services is $U(1500000,4500000)$ [9].

(4) Evolutionary algorithm parameters: the population size is 100, the probability of crossover is set as 1.0, the probability of mutation is set as 0.1, and the number of iterations is set as 200.

Furthermore, the Pareto optimum set, which is a collection of non-dominated solutions, is the output result of the fog service placement optimization algorithm based on NSGA-II. We assign a weight to each objective function in the multi-objective optimization problem and then sum them to choose the optimal individual from the Pareto optimal set for experimental analysis. The sum value is taken as the criterion to select the best individual from the Pareto optimal set. The general form is as follows:

$$f = \sum_{\Omega} w_i \times f_i(x) \tag{5-1}$$

where $w_i \in [0,1]$ is the weight, and Ω denotes the number of objective functions. $f_i(x)$ is the value of the objective function.

Each individual objective function value needs to be normalized before using formula (5-1), and the normalized objective function value is bring into the formula 5-1 to calculate the fitness value of the individual. Combining formulas 3-9, the value of the optimization objective $1 - \text{Resource Usage}$ is between 0 and 1, so it does not need to be normalized. The formula of normalization of Cost and Latency is as follows:

$$f'_i(x) = \begin{cases} \dfrac{f_i(x) - f_i^{min}(x)}{f_i^{max}(x) - f_i^{min}(x)}, & f_i^{max}(x) \neq f_i^{min}(x) \\ 0, & f_i^{max}(x) = f_i^{min}(x) \end{cases}$$

where $f_i(x)$ is the objective function value (Cost or Latency), $f_i^{max}(x)$ is the largest $f_i(x)$ in the current population size, and $f_i^{min}(x)$ is the smallest $f_i(x)$ in the current population size.

It is worth noting each optimization objective is assigned the same weight of 1/3.

5.2. Analysis of experimental results

The effect of population size on the proposed scheme

We explored the effects of various population sizes on the experimental results first. In order to reduce the error of the experimental data, the experimental data are the average values after running 10 times in this article. Figure 4 shows the change of the sum of the weights of the optimal individuals in the Pareto optimal set in each iteration when the population size is 50, 100, and 150, respectively. The larger the population size, the fewer iterations the algorithm reaches convergence. This paper focuses on the minimization process, so the sum of the weights is proportional to the optimization effect.

Figure 5 shows the variation in the time required for each iteration of the scheme proposed in this paper with the number of iterations when the population size is 50, 100, and 150 respectively. The time required in each iteration will increase as the population size increases. This is due to the time complexity of NSGA-II is $O(MN^2)$ [22], where M is the number of optimization objectives and N is the size of the population. The weight sum value of the optimal individual and the iteration time of the algorithm are both good when the population size is 100, as shown in Figs. 5.1 and 5.2, so the population size is set to 100 in the next experiments.

Scheme comparison

In order to verify the performance of our scheme, we use the weighted sum genetic algorithm (WSGA) to optimize the multi-objective fog service placement problem constructed in this paper. This solution is the most widely used optimization algorithm to solve multi-objective optimization problems in existing studies [12][25]. The main idea is as follows: each objective function in the multi-objective optimization problem is assigned a weight, and the weight sum is used as the individual scalar fitness value. In the following, we compare the NSGA-II algorithm with the WSGA algorithm.

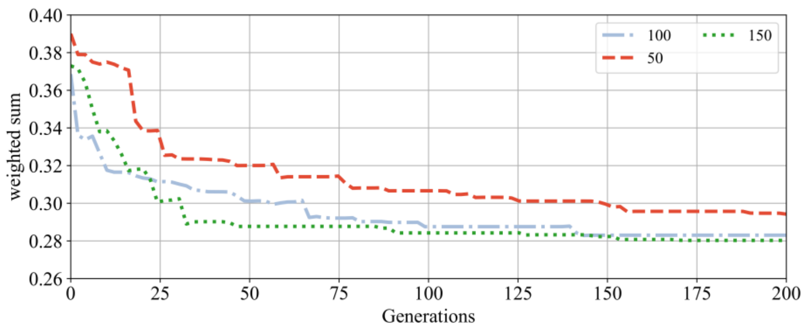

Figure 4. The effect of population size on the sum of weights

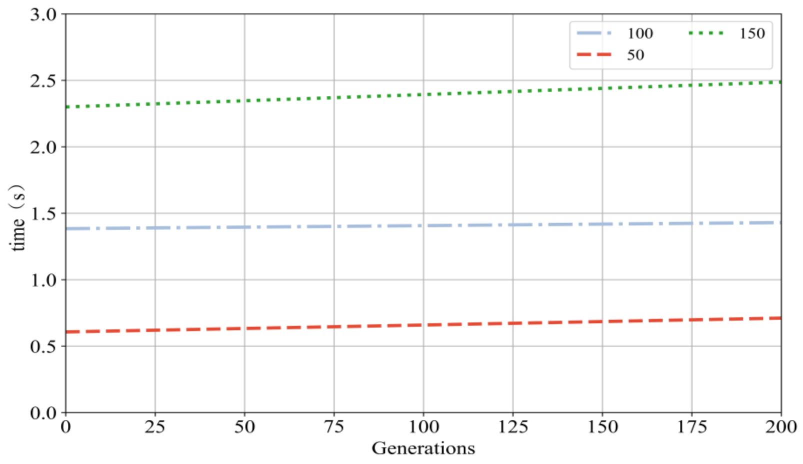

Figure 5. The effect of the population size on the iteration time

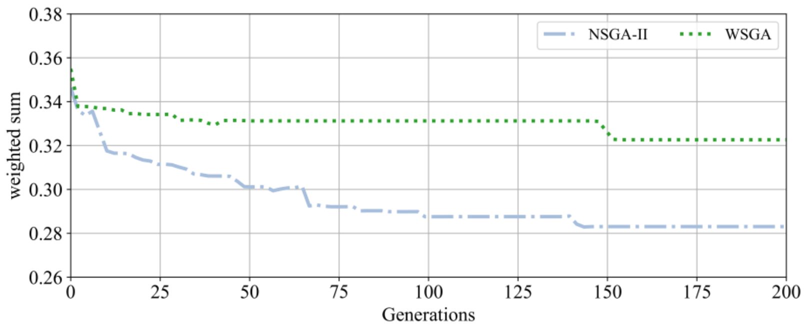

Figure 6. Comparison of the weighted sum value

Figure 6 shows the change of the weighted sum value of the optimization objectives of the two optimization algorithms in optimal schemes under different iterations. The optimization target weighted sum value of the optimal solution changes more slowly and requires the most iterations to converge for the WSGA algorithm. As for NSGA-II algorithm, the weighted sum value of the optimization objective of the optimal scheme is the minimum when the algorithm converges. Therefore, from the perspective of weighted sum value of optimization objective of the optimal scheme, NSGA-II algorithm has better performance than WSGA algorithm.

In order to further analyze the two optimization algorithms, we compare the three objective function values of the optimal scheme in each iteration. Figure 7, Figure 8, and Figure 9 respectively show the changes in idle resource utilization, service cost, and network delay of the optimal solution selected by the two optimization algorithms in each iteration. First, the solution proposed in this paper is much better than the WSGA method in terms of the system's idle resource rate. It can be seen from formula 3-10 that the idle resource rate of the system is inversely proportional to the resource utilization of the system. Therefore, the NSGA-II algorithm precedes the WSGA algorithm in terms of system resource utilization rate.

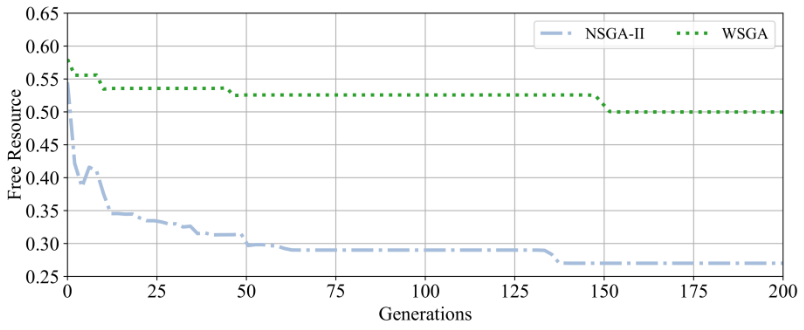

Figure 7. Comparison of free resource ratio

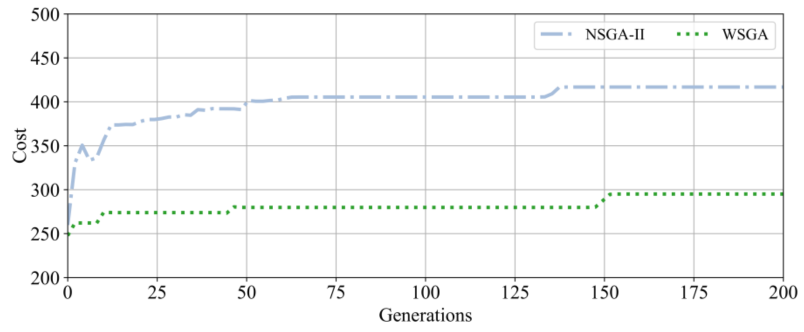

Figure 8. Comparison of service costs

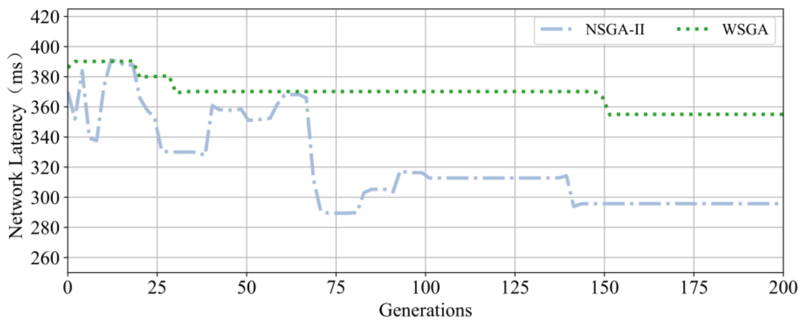

Figure 9. Comparison of network latency

Secondly, the optimization algorithm proposed in this paper is worse than the WSGA algorithm from the perspective of service cost. This is because the amount of resources owned by each fog device remains constant during the evolution process. That is, the denominator in formula 3-9 is constant, so the resource utilization rate of the system is proportional to the total amount of resources required to instantiate deployed services. For the calculation of service cost, we assumed the number of services on the cloud server remains constant in system during the evolution process. That is, the service placement cost remains constant on the cloud server. According to formula 3-11, the service cost is proportional to the total amount of resources required to instantiate deployed services. Therefore, the change trend of system resource utilization and service cost is consistent. Only the cost of placing service instances on the fog device is considered in this article, while the communication cost and calculation cost of the service are ignored, resulting in the above situation. We will build a more complex service cost model in future work.

From the perspective of network delay, the optimization algorithm proposed is better than WASG algorithm. As show in Fig. 5.6, in the early stages of the iteration, the network delay of the optimal solution selected by the NSGA-II algorithm is extremely irregular. However, the optimal individuals selected from this algorithm have the smallest network latency in most cases.

Based on Fig. 5.3, Fig. 5.4, Fig. 5.5 and Fig. 5.6, the performance of the WSGA algorithm is relatively poor, and only the service cost is better than the other two optimization algorithms. The NSGA-II algorithm achieves the highest optimization goal, but the service cost of the algorithm is higher than WSGA algorithm. In addition, the NSGA-II algorithm has a faster convergence rate than the WSGA algorithm.

6. Conclusion

This paper proposes an optimized service placement scheme in a fog computing environment.The goal of the algorithm is to increase the resource utilization of the fog layer as much as possible, reduce the network latency, and reduce the cost of service placement. Considering that the service placement problem in the fog computing environment is NP-hard, an evolutionary algorithm named NSGA-II is applied to optimize the solution of multi-objective FSPP. The experimental results show that the solution proposed in this paper has excellent performance compared with the existing solutions. The experimental results show that the solution proposed in this paper has better performance than existing solutions. In future work, we will build a more complex mathematical model of service placement costs, which includes not only the cost of service placement, but also the storage, communication and computing costs of services. In addition, using other meta-heuristic algorithms (MOA/D, PSO, etc.) to optimize multi-objective service placement schemes is one of the future research directions.

References

[1] Bonomi F, Milito R, Natarajan P, et al. Fog computing: A platform for internet of things and analytics[M]//Big data and internet of things: A roadmap for smart environments. Springer, Cham, 2014: 169-186.
[2] Pasteris S, Wang S, Herbster M, et al. Service placement with provable guarantees in heterogeneous edge computing systems[C]//IEEE INFOCOM 2019-IEEE Conference on Computer Communications. IEEE, 2019: 514-522.
[3] Velasquez K, Abreu D P, Curado M, et al. Service placement for latency reduction in the internet of things[J]. Annals of Telecommunications, 2017, 72(1-2): 105-115.
[4] Yousefpour A, Patil A, Ishigaki G, et al. FogPlan: a lightweight QoS-aware dynamic fog service provisioning framework[J]. IEEE Internet of Things Journal, 2019, 6(3): 5080-5096.
[5] Barcelo M, Correa A, Llorca J, et al. IoT-cloud service optimization in next generation smart environments[J]. IEEE Journal on Selected Areas in Communications, 2016, 34(12): 4077-4090.
[6] Bittencourt L F, Diaz-Montes J, Buyya R, et al. Mobility-aware application scheduling in fog computing[J]. IEEE Cloud Computing, 2017, 4(2): 26-35.
[7] Souza V B C, Ramírez W, Masip-Bruin X, et al. Handling service allocation in combined fog-cloud scenarios[C]//2016 IEEE international conference on communications (ICC). IEEE, 2016: 1-5.
[8] Dapeng W U, Ji L Y U, Zhidu L I, et al. Mobility aware edge service migration strategy[J]. Journal on Communications, 2020, 41(4): 1.
[9] Lera I , Guerrero C , Juiz C . Availability-aware Service Placement Policy in Fog Computing Based on Graph Partitions[J]. IEEE Internet of Things Journal, 2018:1-1.
[10] Urgaonkar R, Wang S, He T, et al. Dynamic service migration and workload scheduling in edge-clouds[J]. Performance Evaluation, 2015, 91: 205-228.
[11] Moallemi R, Bozorgchenani A, Tarchi D. An Evolutionary-based Algorithm for Smart-living Applications Placement in Fog Networks[C]//2019 IEEE Globecom Workshops (GC Wkshps). IEEE, 2019: 1-6.
[12] Skarlat O, Nardelli M, Schulte S, et al. Optimized IoT service placement in the fog[J]. Service Oriented Computing and Applications, 2017, 11(4): 427-443.
[13] Ni L, Zhang J, Jiang C, et al. Resource allocation strategy in fog computing based on priced timed petri nets[J]. ieee internet of things journal, 2017, 4(5): 1216-1228.
[14] Canali C, Lancellotti R. GASP: genetic algorithms for service placement in fog computing systems[J]. Algorithms, 2019, 12(10): 201.
[15] Yang L, Cao J, Liang G, et al. Cost aware service placement and load dispatching in mobile cloud systems[J]. IEEE Transactions on Computers, 2015, 65(5): 1440-1452.
[16] Zhang B , Wang X , Huang M . Multi-objective Optimization Controller Placement Problem in Internet-oriented Software Defined Network[J]. Computer Communications, 2018, 123(JUN.):24-35.
[17] Natesha B V , Guddeti R . Adopting elitism-based Genetic Algorithm for minimizing multi-objective problems of IoT service placement in fog computing environment[J]. Journal of Network and Computer Applications, 2021.
[18] Balalaie A, Heydarnoori A, Jamshidi P. Microservices architecture enables devops: Migration to a cloud-native architecture[J]. Ieee Software, 2016, 33(3): 42-52.
[19] Lera I, Guerrero C, Juiz C. Availability-aware service placement policy in fog computing based on graph partitions[J]. IEEE Internet of Things Journal, 2018, 6(2): 3641-3651.
[20] Salaht F A, Desprez F, Lebre A. An overview of service placement problem in fog and edge computing[J]. ACM Computing Surveys (CSUR), 2020, 53(3): 1-35.
[21] Yousefpour A, Patil A, Ishigaki G, et al. FogPlan: a lightweight QoS-aware dynamic fog service provisioning framework[J]. IEEE Internet of Things Journal, 2019, 6(3): 5080-5096.
[22] K, Pratap A, Agarwal S, et al. A fast and elitist multiobjective genetic algorithm: NSGA-II[J]. IEEE transactions on evolutionary computation, 2002, 6(2): 182-197.
[23] Mayer R, Graser L, Gupta H, et al. Emufog: Extensible and scalable emulation of large-scale fog computing infrastructures[C]//2017 IEEE Fog World Congress (FWC). IEEE, 2017: 1-6.
[24] Yousefpour A, Patil A, Ishigaki G, et al. FogPlan: a lightweight QoS-aware dynamic fog service provisioning framework[J]. IEEE Internet of Things Journal, 2019, 6(3): 5080-5096.
[25] Wen Z, Yang R, Garraghan P, et al. Fog orchestration for internet of things services[J]. IEEE Internet Computing, 2017, 21(2): 16-24.
[26] Velasquez K, Abreu D P, Curado M, et al. Service placement for latency reduction in the internet of things[J]. Annals of Telecommunications, 2017, 72(1-2): 105-115.
[27] Huang X, Ganapathy S, Wolf T. Evaluating algorithms for composable service placement in computer networks[C]//2009 IEEE International Conference on Communications. IEEE, 2009: 1-6.