Hindawi
Wireless Communications and Mobile Computing
Volume 2022, Article ID 2136933, 12 pages
https://doi.org/10.1155/2022/2136933

Research Article

Multiobjective Optimization Model and Algorithm Based on Differential Brain Storm for Service Path Constructing

Xiaoqiang Jia, Li Ge, Yunfei Li, Jun Liu, and Biying Zhou

School of Computer and Technology, Weinan Normal University, Weinan, Shaanxi 714000, China

Correspondence should be addressed to Xiaoqiang Jia; xqjia@wnu.edu.cn

Received 14 March 2022; Revised 12 April 2022; Accepted 5 May 2022; Published 10 June 2022

Academic Editor: Xingsi Xue

Network function virtualization (NFV) can provide the resource according to the request and can improve the flexibility of the network. It has become the key technology of the next-generation communication. Resource scheduling for virtual network function service chain (VNF-SC) mapping is the key issue of the NFV. Aiming at previous research primarily focused on constructing service paths with a single objective, for example, latency minimization, cost minimization, or load balance, which ignored the overall performance of constructed service paths, a multiobjective model, which minimizes benefit, expense, time delay, and load balance, to solve the service path constructing problem, is established. In this model, VNF-SCs are divided into two classes, i.e., part of the required VNFs in each VNF-SC is dependent, and others are independent. To solve this multiobjective model, an algorithm based on discrete difference brain storm (MO2DBS) in the framework of MOEA/D was proposed. In the new algorithm, a two dimensional integer coding is designed. In addition, a difference mutation was used to replace the original Gaussian mutation to adjust the mutation step adaptively. Simulation experiments show that the proposed algorithms can obtain higher benefit, load balance, lower expenses, and time delay than the compared algorithms.

1. Introduction

Network function virtualization (NFV) [1–3] proposes the concept of "softwarization," which decouples network functions from customized hardware devices and uses virtualization technology to build software units. Through the deployment and combination of VNF, diversified service customization can be realized. Compared with the traditional middlebox approach, NFV enables flexible development, deployment, and migration of network functions, thus effectively reducing the cost of equipment investment. Combined with SDN (software defined network) [4, 5] technology, the management program on the upper layer of the controller is adopted to control the network, which further improves the manageability and reduces the operation and maintenance cost [6, 7].

In NFV architecture, in order to meet diversified application requirements and realize on-demand service customization, the controller needs to implement the service routing algorithm to map each service in the demand to the node that can carry the service in order to build an end-to-end data path, so that the network flow is processed by the required service in turn [8]. This kind of directly connected sequence composed of nodes and links is called service path, which traverses the whole network to meet specific function and performance requirements. This problem is called service path construction problem, and the ordered service sequence contained in application request is called service chain. In the process of constructing the service path, the following problems need to be solved [9]: (1) the execution of the service requires computing and storage resources, and the transmission of data requires bandwidth resources. Under the condition of limited resource capacity, how to allocate resources efficiently to map more service chains. (2) Service providers need to pay for rented resources to build service paths according to the users' demands and provide customized services to users through the service paths to obtain benefits. Therefore, costs and benefits need to be included in the evaluation system, and how to effectively rent

resources should be considered to reduce costs and improve benefits. (3) A service instance can be deployed on any network node. The location of the service bearer node affects the transmission delay and link resource usage of the service path, the performance affects the processing delay of the service instance, and the available resources affect the capacity of carrying the service. Therefore, it is necessary to solve the problem of how to choose among many candidate service instances and construct the optimal service path [10, 11].

Since service path construction is an NP-hard problem [12–14], the research work has been mainly focused on designing heuristic algorithms to obtain approximate optimal solutions. Literature [15] proposed a layered graph to solve the problem of service execution sequence. The method of constructing service step search graph, which ensured the service execution sequence and also took into account the problem of excessive use of resources, is proposed [16]. Literature [17] used the reciprocal of the available resource capacity to reset the weights of hierarchical graph edges and proposed a load balancing algorithm. Literature [18] proposed a heuristic algorithm based on dynamic programming to achieve the balance between cost and performance through the dynamic orchestration of VNF. Literature [19] proposed a mapping algorithm based on service chain decomposition, so that VNF based on the same implementation technology can be preferentially interconnected and mapped to the same server, thus effectively reducing the overall cost. Literature [20] maps topology fragments to candidate servers in data centers by requesting topology segmentation and uses virtual gateways to construct service chain diagrams to realize traffic aggregation, so as to solve cross-domain service chain mapping problems. Literature [21] studied the mechanism of network function division, description, and combination under SDN architecture and proposed a node-first algorithm. This literature builds the security service path through the flexible combination of VNF, so as to provide customized security services.

These studies have been explored from different perspectives, but they may be aimed at minimizing overhead, minimizing end-to-end delay, or load balancing. Few studies have been able to comprehensively consider the construction of a service path that not only optimizes delay and cost but also balances load. Aiming at previous research primarily focused on constructing service paths with a single objective, for example, latency minimization, cost minimization, or load balance, which ignored the overall performance of constructed service paths, a multiobjective model, which minimizes benefit, expense, time delay, and load balance, to solve the service path constructing problem, is established. In this model, VNF-SCs are divided into two classes, i.e., part of the required VNFs in each VNF-SC is dependent, and others are independent. To solve this multiobjective model, an algorithm based on discrete difference brain storm (MO2DBS) in the framework of MOEA/D was proposed. In the new algorithm, a two dimensional integer coding is designed. In addition, a difference mutation was used to replace the original Gaussian mutation to adjust the mutation step adaptively. Simulation experiments show that the proposed algorithms can obtain higher benefit, load balance, lower expenses, and time delay than the compared algorithms.

2. Multiobjective Service Path Building Model

2.1. Network Model

2.1.1. Physical Network. The topology of the physical network can be described as a weighted undirected graph, denoted as $G^S = (N^S, L^S, A_N^S, A_L^S)$, where N^S and L^S denote the set of nodes and links in the network, respectively. A_N^S denotes the resource property of node n^S in N^S, including the available CPU capacity $C(n^S)$ of the node and the service set $S(n^S) = S_k$ that can be provided by the node n^S. The processing time of the service s_k in N^s denoted as $d_{s_k}(n^s)$, per time unit CPU capacity, is c^S. A_N^S denotes the resource property of the link, including the link available bandwidth $B(l^S)$, transmission delay $D(l^S)$, and unit bandwidth resource cost b^S. The set of all acyclic paths in the physical network is represented as Π^S. The set of acyclic paths between any two nodes n_i^S and $n_j^S \in N^S$ is marked as $\Pi^S(n_i^S, n_j^S)$. p^S represents a acyclic path in the physical network, and $H(p^S)$ denotes the length of path p^S, i.e., hop number. As shown in Figure 1, S1~S4 in physical network nodes represent the services that nodes can instantiate, the numbers inside nodes represent the current available computing capacity of nodes, and the numbers on links represent available bandwidth and transmission delay.

2.1.2. Service Request. A service request represents a user's specific functional and resource requirements in the form of a logical service path, whose topology can be described as a weighted directed graph labeled as $G^R = (N^R, L^R, R_N^R, R_L^R)$. The directivity of an edge represents a constraint on the execution order of a service instance. $N^R = \{n_0^R, n_1^R, \cdots, n_c^R, n_{c+1}^R\}$ represents a collection of logical path nodes. n_0^R and n_{c+1}^R denote the source node and the destination node, respectively. c represents the number of services required. $L^R = \{<n_0^R, n_1^R>, <n_0^R, n_1^R>, \cdots, <n_c^R, n_{c+1}^R>\}$ denotes a set of logical links. R_N^R denotes the collection of requirement attributes of node n_i^R, service needs $S(n_i^R)$, computing capacity requirements $\mu(S(n_i^R))$, the rental unit computing capacity required to pay fees c^R, etc. Similarly, R_L^R denotes the collection of requirement attributes of link l^R; it consists of the bandwidth requirement (l^R) and the charge for renting unit bandwidth b^R. Each service request can be represented by the triplet $SR(G^R, t_a, t_d)$, where t_a indicates the arrival time of the request and t_d indicates the duration of the request. As shown in Figure 1, for service request SR1, A and G represent the source node and destination node, respectively, S1~S3 within logical nodes, A, B, and C represent the required service, the numbers inside the nodes represent the computing power demand, A~G represent the underlying network nodes, and the numbers on the logical link represent the bandwidth required for data transmission. In addition, the dotted arrows in Figure 1 represent the mapping of services. Dashed lines between A and G indicate that there is no mapping between nodes.

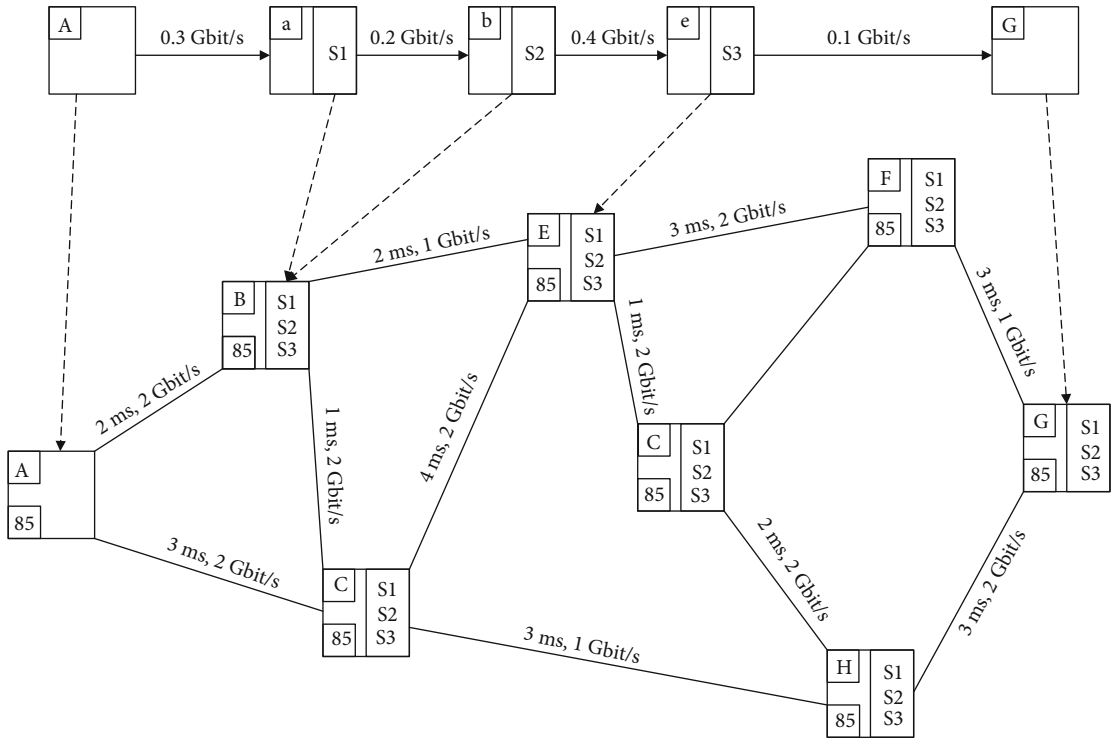


Figure 1: Examples of physical network and service paths.

2.2. Formula Description. $SC = S(n_1^R), S(n_2^R), \cdots, S(n_c^R)$ denotes the service chain required in the service request; each remaining logical node represents a required service except the source node and the destination node. Service path to complete the build process can be described as after receiving the user's service request, the service is mapped to the controller; in turn, SC can provide the service and computing ability to meet the demand of the underlying nodes. At the same time, in the service load between the nodes to establish and meet the corresponding optimal underlying path bandwidth demand, it ensures that the data flows on the path pass through the service instances agreed on the SC and finally build an end-to-end service path that meets the requirements of specific functions, performance, and latency. If the path does not exist, the request is rejected.

2.2.1. Objective Function. A single service path built for a service request needs to achieve the following goals:

(1) Profit. Similar to literature [5, 13], the revenue obtained by the service provider for successfully constructing a service path for a service request is equal to the sum of the fees paid by the users for renting computing resources and bandwidth resources, as defined below:

$$R(SP) = \sum_{n^r \in N^R} \mu\left(S\left(n^R\right)\right) c^R + \sum_{l^r \in L^R} \mu\left(l^R\right) b^R. \quad (1)$$

The cost paid by a service provider to successfully construct a service path for a user is equal to the sum of the computing resources of the underlying node and the bandwidth resources of the underlying link, as defined below:

$$C(SP) = \sum_{n^r \in N^R} \mu\left(S\left(n^R\right)\right) c^R + \sum_{l^r \in L^R} \sum_{P^S \in M_L\left(l^R\right), l^s \in P^S} \mu\left(l^R\right) b^R. \quad (2)$$

The profit of the service request is the revenue minus to the cost; thus, the profit can be defined as

$$P(SP) = \sum_{l^r \in L^R} \mu\left(l^R\right) b^R - \sum_{l^r \in L^R} \sum_{P^S \in M_L\left(l^R\right), l^s \in P^S} \mu\left(l^R\right) b^R. \quad (3)$$

Since $P(SP) < R(SP)$, thus, we can normalise the objective as follows:

$$\min f_1 = \min \left\{ \frac{P(SP)}{R(SP)} \right\}. \quad (4)$$

According to the definition, we can see that $0 \le f_1 \le 1$.

(2) Transmission Delay. The end-to-end delay of the service path represents the time taken by the data flow from the source node to the destination node and is composed of the execution delay of the service instance on the service path and the transmission delay of the communication link, as defined below:

$$D(SP) = \sum_{n^R \in N^R} \sum_{n^S \in M_S(S(n^R))} d_{S(n^R)} - \sum_{l^r \in L^R} \sum_{P^S \in M_L(l^R), l^S \in P^S} d\left(l^S\right). \tag{5}$$

Another variable is defined:

$$D'(SP) = \sum_{n^R \in N^R} \sum_{n^S} d_{S(n^R)} - \sum_{l^r \in L^R} \sum_{l^s \in P^S} d\left(l^S\right). \tag{6}$$

Obviously, we have $D(SP) < D'(SP)$; thus, $0 \le f_2 = D(SP)/D'(SP) \le 1$. The second objective is defined as

$$\min f_2 = \min \left\{ \frac{D(SP)}{D'(SP)} \right\}. \tag{7}$$

(3) Load Degree. When constructing the service path, besides the functional requirements of the service, the load of the service bearing node and its adjacent links should be considered to map the service and logical links to the resource-rich underlying nodes and links as far as possible. Load degree (LD) measures the resource usage and load of a service path. The definition of the load intensity of the bottom node $n_{i_k}^R$ in the service path is as follows:

$$LD_{n_{i_k}^R} = \sum_{S(n^R) a n_{i_k}^R \in M_{i_k}} \frac{\mu\left(S\left(n^R\right)\right)}{C\left(n_{i_k}^R\right)}, \forall n_{i_k}^R \in N^S P. \tag{8}$$

The load intensity of a node takes into account the available computing capacity of the node and the computing capacity demand of the service. According to Formula (3), its value ranges from 0 to 1. The smaller the value is, the more likely it is to map services to this node, and the more beneficial it is to load balancing of the underlying node. Based on the idea of node load intensity, the load intensity of link Sl in the service path is defined as follows:

$$LD_{l_{i_k}^R} = \sum_{P^S \in M_L\left(l^R\right), l_{i_k}^R \in P^S,} \frac{\mu\left(l^R\right)}{B\left(l_{i_k}^R\right)}, \forall l_{i_k}^R \in L^S P. \tag{9}$$

The load intensity of the service path SP is defined as

$$LD(SP) = w_N \sum_{n_{i_k}^S \in N^{SP}} LD_{n_{i_k}^S} + w_L \sum_{l_{i_k}^S \in L^{SP}} LD_{l_{i_k}^S}, \tag{10}$$

where w_N and w_N are the two weight parameters used to adjust the load intensity of nodes and links, and $0 \le w_N, w_N \le 1$ and $w_N + w_N = 1$. Similarly, another variable is defined:

$$LD'(SP) = \sum_{n_{i_k}^S \in N^{SP}} LD_{n_{i_k}^S} + \sum_{l_{i_k}^S \in L^{SP}} LD_{l_{i_k}^S}. \tag{11}$$

Obviously, we have $LD(SP) < LD'(SP)$; thus, $0 \le f_3 = LD(SP)/LD'(SP) \le 1$. The third objective is defined as

$$\min f_3 = \min \left\{ \frac{LD(SP)}{LD'(SP)} \right\}. \tag{12}$$

2.2.2. Constraints

(1) For computing resource constraints, ensure that the resources available on the underlying nodes can meet the needs of executing the service instance:

$$\sum_{u \in N^R} x_i^{S(u)} \mu(S(u)) \le C(i), \forall i \in N^{SP}. \tag{13}$$

The $x_i^{S(u)}$ value is 1 if the service $S(u)$ required by the logical node u is mapped to the underlying node i, and otherwise is 0

(2) Bandwidth resource constraints ensure that the bandwidth available on the underlying link is sufficient to bear the logical link mapped to

$$\sum_{l_{v,u} \in L^R} y_{ij}^{uv} \mu(l_{u,v}) \le B\left(l_{i,j}\right), \forall l_{i,j} \in L^{SP}. \tag{14}$$

If the underlying link $L_{i,j}$ carries the logical link $L_{u,v}$, the y_{ij}^{uv} value is 1. Otherwise, the value is 0

(3) The service-to-underlying node mapping constraint ensures that services in the same service request can be mapped to only one underlying node.

$$\sum_{i \in N^S} x_i^{S(u)} = 1, \forall u \in N^R \tag{15}$$

(4) The underlying node hosts constraints on services, and one underlying node can host multiple services for the same service request.

$$\sum_{u \in N^R} x_i^{S(u)} \le \lambda, \forall u \in N^{SP} \tag{16}$$

(5) Connectivity constraints on logical links to mapped underlying network paths:

$$\sum_{L_{i,j}\in L^{SP}} y_{ij}^{uv} - \sum_{L_{ji}\in L^{SP}} y_{ji}^{uv} = x_i^{S(u)} - x_i^{S(v)}, \forall i \in N^{SP} \quad (17)$$

(6) End-to-end latency constraint ensures that the end-to-end latency of service paths meets the service level agreement (SLA) requirements.

$$\sum_{u\in N^R} \sum_{i\in N^{SP}} x_i^{S(u)} d_{S(u)(i)} + \sum_{l_{u,v}\in L^R} \sum_{L_{i,j}\in L^{SP}} y_{ij} d_{l_{i,j}} \le D_{\max} \quad (18)$$

(7) Variable's constraints:

$$\begin{aligned} &\forall i \in N^S, \\ &u \in N^R, \\ &x_i^S(u) \in \{0, 1\} \\ &\forall L_{i,j} \in L^S, \\ &L_{u,v} \in L^R, \\ &y_{ij}^{uv} \in \{0, 1\} \end{aligned} \quad (19)$$

3. Multiobjective Discrete Difference Brain Storm (MO2DBS)

In recent years, BSO, as a new swarm intelligence optimization algorithm, has attracted extensive attention of many scholars. BSO and its improved version have been successfully applied in some fields, for example, satellite formation optimization [22], BRUShless DC motor optimization [23], image processing [24], and route planning [25]. On the basis of BSO, scholars at home and abroad derived different brainstorming optimization algorithms based on its clustering, selection, mutation, and other methods, which improved the performance of the algorithm [26]. BSO has the following four methods to select individuals to be mutated: (1) select a class according to roulette probability, and select the class center of the class as individuals to be mutated. (2) A class was selected according to the roulette probability, and a random individual in the class was selected as the individual to be mutated. (3) Two classes were randomly selected, and the class centers of the two classes were fused to become individuals to be mutated. (4) Two classes were randomly selected, and one individual was randomly selected from each of the two classes and fused into the individual to be mutated. Select the fusion process in the operation by pressing the following formula:

$$y = r \cdot x_1 + (1 - r) \cdot x_2, \quad (20)$$

where y is the individual to be mutated after the fusion of two individuals, x_1 and x_2 are the two individuals receiving fusion, and r is a random number from 0 to 1 that adjusts the weight of two individuals.

The mutation operation will add disturbance quantity to the individual to be mutated, which is called the mutation step size in this paper. Gaussian mutation is carried out as follows:

$$y^d = x^d + \xi \cdot N(\mu, \sigma), \quad (21)$$

where y^d is the d-dimension of the new individual, x^d is the d-dimension of the individual to be mutated, and $N(\mu, \sigma)$ is a Gaussian random number whose mean value is ¦ I and variance is ¦ O. The calculation formula of coefficient of variation is as follows:

$$\xi = \log sig\left(\frac{0.5e_{\max} - e}{k}\right) \times \text{rand}\,(), \quad (22)$$

where $e_{\max}$ is the maximum number of iterations, e is the current iteration number, k is the coefficient regulating the slope of the S-type transfer function log $sig()$, and rand () is a random number between 0 and 1.

At the beginning of human brainstorming, everyone's ideas are very different. When creating new ideas, take into account the differences between the existing ideas. Therefore, difference variation is used to determine the variation step. The mutation operation based on differential variation is as follows:

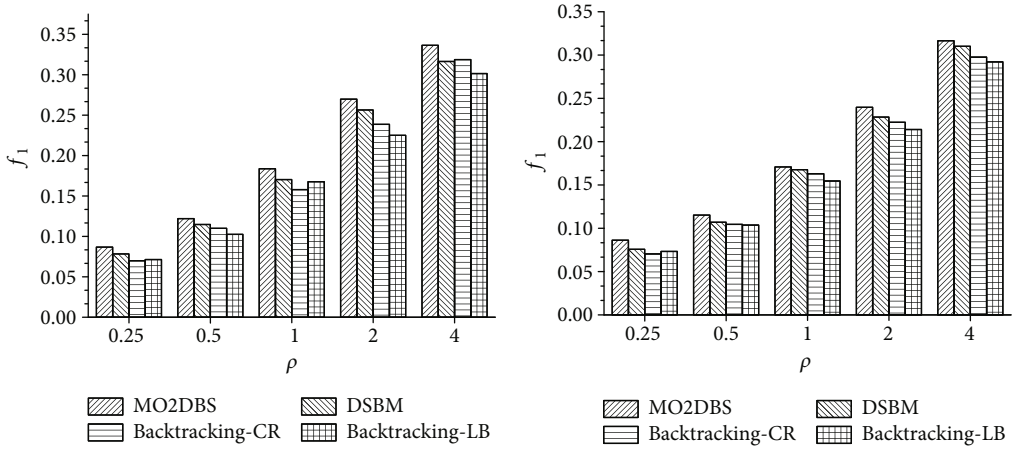
$$y = \begin{cases} R \times (H_d - L_d) + L_d, & \text{rand}\,() < p_r, \\ x + R. \times (x_a - x_b), & \text{else}, \end{cases} \quad (23)$$

where L_d and H_d are the boundary values of the search space, p_r means that there is a certain probability to get a random new individual, and x_a and x_b are any two different individuals in the population.

Differential variation has two advantages over Gaussian variation. On the one hand, the operation of Gaussian variation includes logsig() function, Gaussian distribution function, random function, and four mixed operations, while difference variation only has random function and four mixed operations, which greatly reduces the amount of computation. On the other hand, the step size of differential variation is based on the contemporary population and adaptively adjusts according to the dispersion degree of individuals in the population: there is a larger step size when the population is dispersed, and a smaller step size when the population is concentrated. The mutation step size is confirmed in real time according to the population feedback, and the algorithm can capture the search feature better.

4. Experiment and Analysis

4.1. Experimental Setting. The underlying network topology is randomly generated by GT-ITM tool, which contains 50 nodes and about 130 links. The computing capacity of the bottom node and the bandwidth of the bottom link are evenly distributed [50,100], and the cost of unit computing



(a) Profit obtained in NSFNET when $N_D = N_V/5$

(b) Profit obtained in CHNNET when $N_D = N_V/5$

Figure 2: Total profit obtained when $N_D = N_V/5$.

resource cS and bandwidth resource bS are 1. The number of service types supported by the underlying network is set to 10. Each underlying node provides one to five service types randomly. According to literature [5, 24], the processing time of a service instance depends on the type of the service and the processing capacity of the network node. The transmission delay of the underlying link is proportional to the Euclidean distance between the two endpoints of the link, which is set according to the above principle and ranges from 1 to 10 time units.

The arrival process of service requests obeys the Poisson distribution, with an average of four requests arriving within 100 time units, and the duration of each service request obeys an exponential distribution with an average of 1000 time units. The service chain is composed of four services, the service type is random and nonrepetitive, the computing power required by each service is evenly distributed in [1, 50], and the bandwidth required by logical link is evenly distributed in [1, 50], and the charges cR and bR for unit computing power and unit bandwidth are both 1. The maximum allowable end-to-end delay Dmax is set to 100 time units. The time of each simulation experiment was about 50000 time units, and the data were recorded every 4000 time units from the 2000 time unit. Each group was set up for 10 simulation experiments, and the experimental results were averaged.

The population size was set as 100, the upper limit of iterations was set as 10000, and the weight parameters w_N and w_L of load intensity in Equation (10) are set as 0.5 and 0.5.

4.2. Experimental Results. To demonstrate the performance of the proposed algorithm, three compared algorithms are introduced. The first literature [27] proposes a novel algorithm to map NSCs to the network infrastructure while allowing possible decompositions of network functions. The algorithm is based on integer linear programming (ILP) which minimizes the cost of the mapping. To solve the scalability issue of the ILP formulation, it targets to minimize the mapping cost by making a reasonable selection of the network function decompositions, represented as DSBM. Similarly, literature [28] proposes a novel backtracking heuristic algorithm for virtual network composition. Based on this algorithm, two approaches with two different objectives are presented. The first approach (Backtracking-CR) was aimed at composing a virtual network using the least amount of network resources, while the second (Backtracking-LB) applies load balancing for virtual network composition. Furthermore, a linear programming approach that optimizes the virtual network composition with an objective of using the least amount of network resources is presented and used to bench mark the heuristic algorithm.

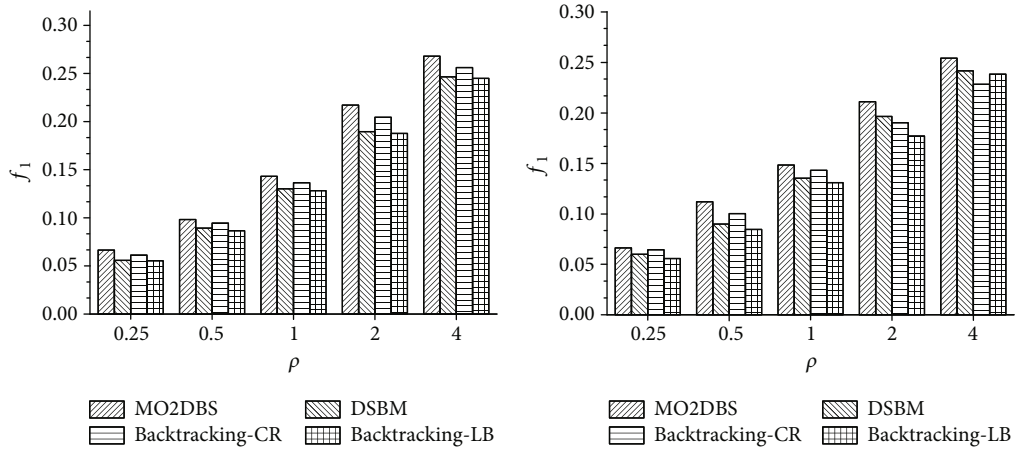
The number of data center nodes is fixed as $N_D = N_V/5$, $N_D = 2N_V/5$, $N_D = 3N_V/5$, and $N_D = 4N_V/5$. In each experiment, number of VNF-SCs is set as $N_R = \rho N_V(N_V - 1)$, and $\rho = 0.25$, 0.5, 1, 2, and 4, respectively. Figures 2–5 show the profit obtained in NSFNET and CHNNET when $N_D = N_V/5$, $N_D = 2N_V/5$, $N_D = 3N_V/5$, and $N_D = 4N_V/5$.

The transmission delay obtained in NSFNST and CHNNET when $N_D = N_V/5$, $N_D = 2N_V/5$, $N_D = 3N_V/5$, and $N_D = 4N_V/5$ is shown in Figures 6–9, respectively.

Similarly, Figures 10–13 show the load degree in NSFNET and CHNNET when $N_D = N_V/5$, $N_D = 2N_V/5$, $N_D = 3N_V/5$, and $N_D = 4N_V/5$.

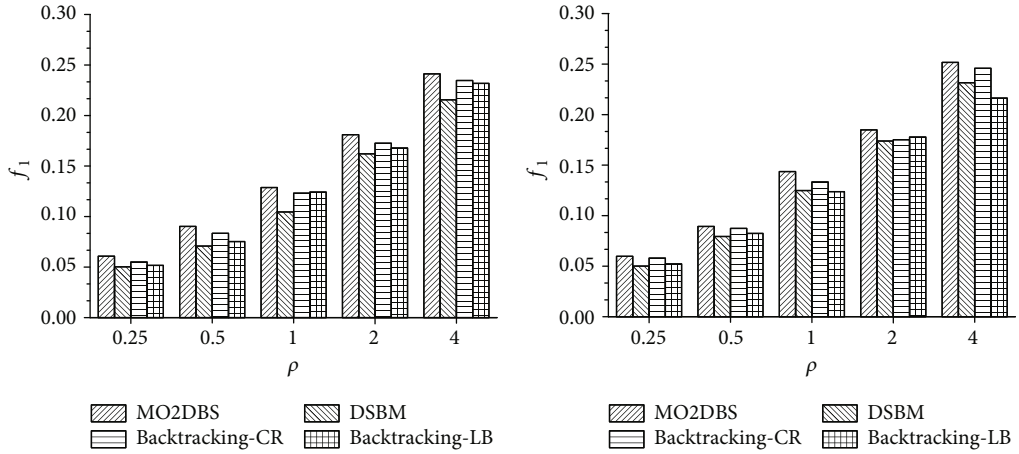
4.3. Experimental Analysis. In simulation experiments, the underlying network builds a service path of long-term average income, and the success rate of average build, average cost, revenue cost ratio, average load intensity, and the end-to-end delay of the main evaluation index were measured and recorded, showing that they change over time, so as to compare different construction strategy of long-term running effect.

Main reason is that when only target path delay, ignoring the underlying network resources and effective use of load balancing, hard to avoid service path through long and resources fragments, leads to excessive consumption of resources, uneven load, and easy generation resource bottleneck, the path of service building success rate and long-term benefits is greatly reduced. And only to load strength



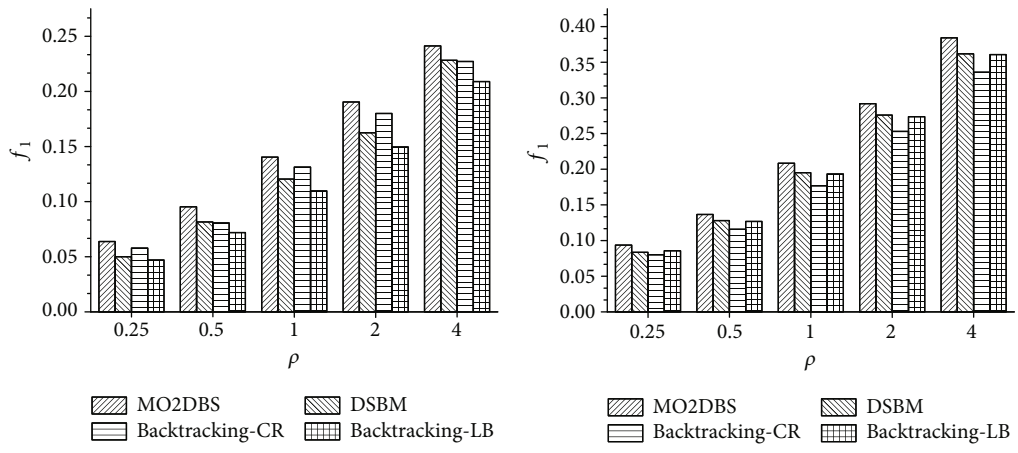
(a) Profit obtained in NSFNET when $N_D = 2N_V/5$ (b) Profit obtained in CHNNET when $N_D = 2N_V/5]$

Figure 3: Total profit obtained when $N_D = 2N_V/5$.



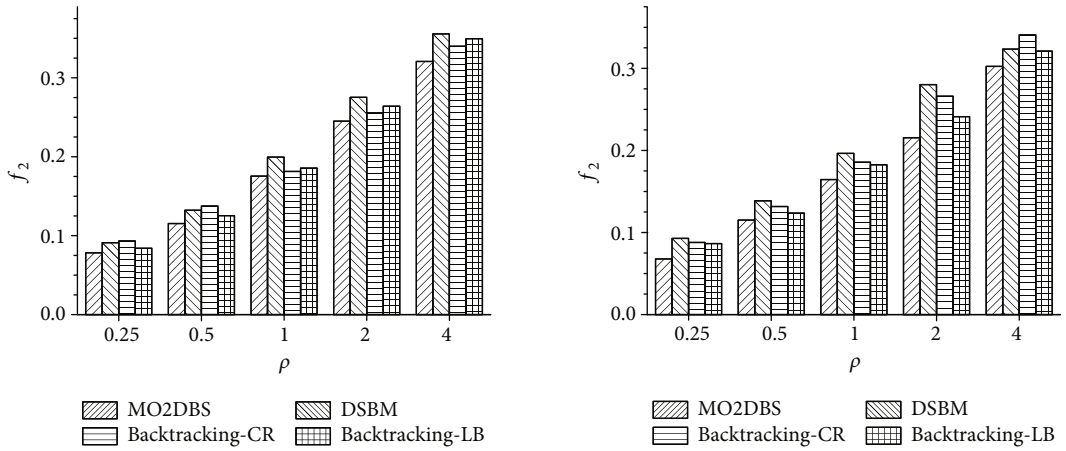
(a) Profit obtained in NSFNET when $N_D = 3N_V/5$ (b) Profit obtained in CHNNET when $N_D = 3N_V/5$

Figure 4: Total profit obtained when $N_D = 3N_V/5$.



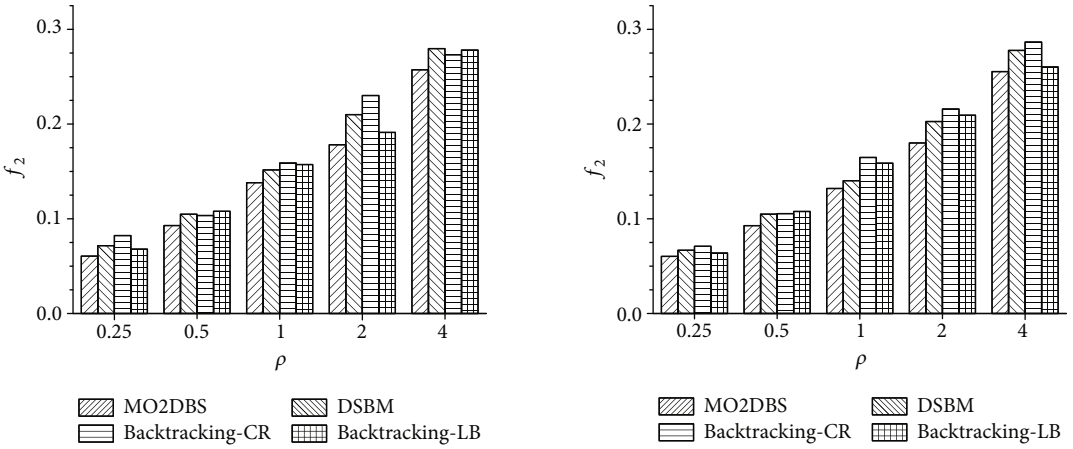
(a) Profit obtained in NSFNET when $N_D = 4N_V/5$ (b) Profit obtained in CHNNET when $N_D = 4N_V/5]$

Figure 5: Total profit obtained when $N_D = 4N_V/5$.



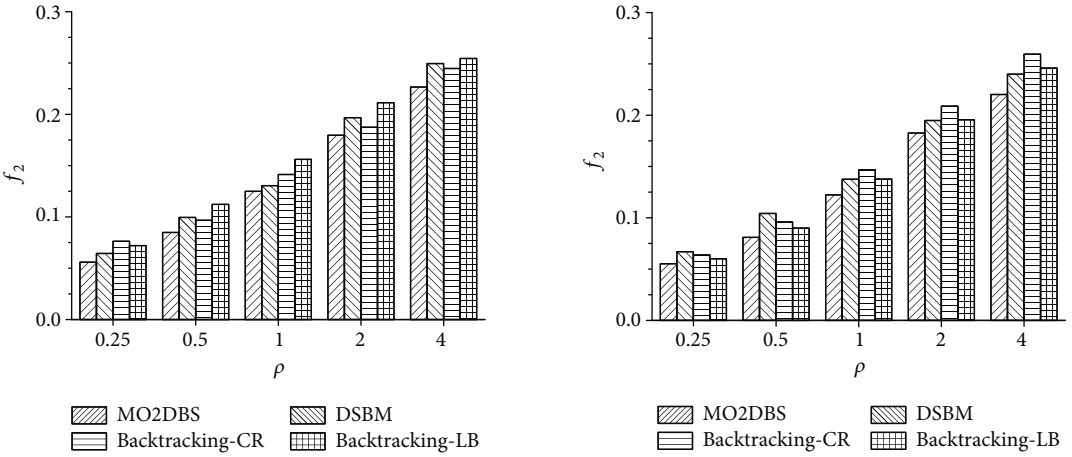
(a) Transmission delay obtained in NSFNET when $N_D = N_V/5$ (b) Transmission delay obtained in CHNNET when $N_D = N_V/5$

Figure 6: Transmission delay obtained when $N_D = N_V/5$.



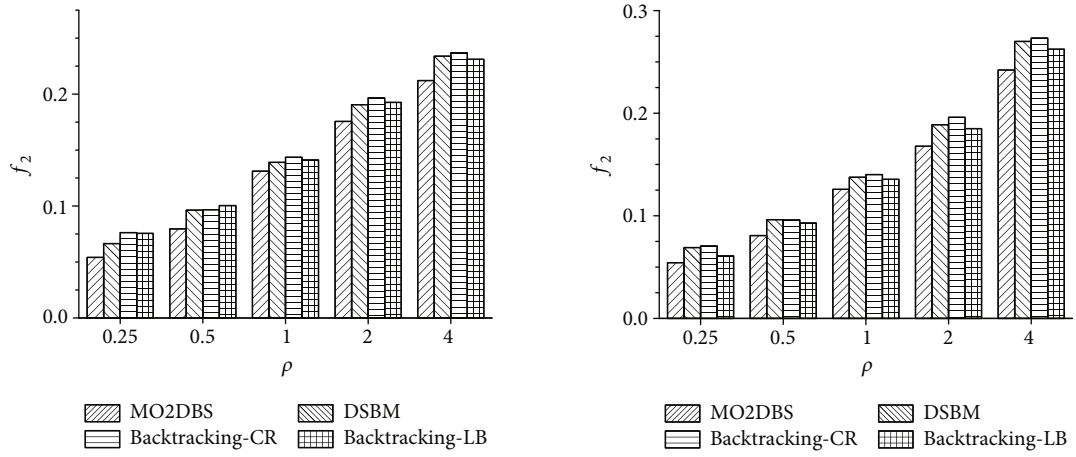
(a) Transmission delay obtained in NSFNET when $N_D = 2N_V/5$ (b) Transmission delay obtained in CHNNET when $N_D = 2N_V/5$

Figure 7: Transmission delay obtained when $N_D = 2N_V/5$.



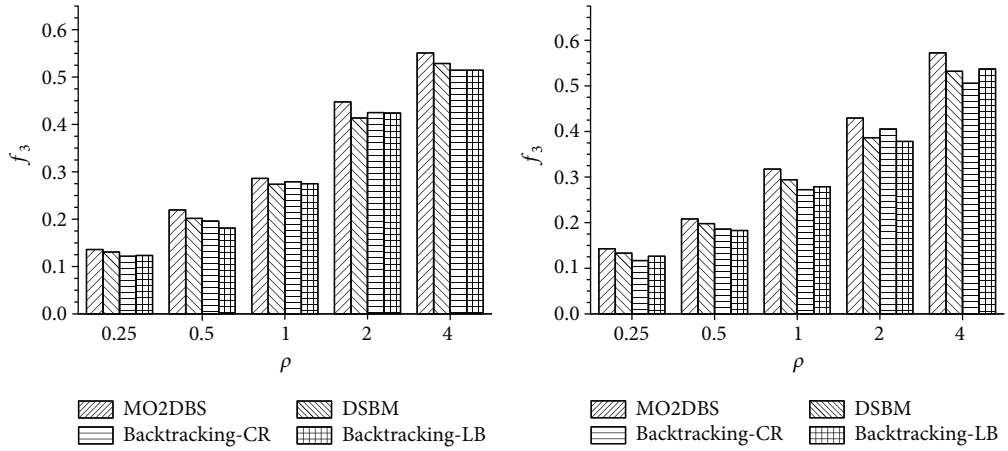
(a) Transmission delay obtained in NSFNET when $N_D = 3N_V/5$ (b) Transmission delay obtained in CHNNET when $N_D = 3N_V/5$

Figure 8: Transmission delay obtained when $N_D = 3N_V/5$.



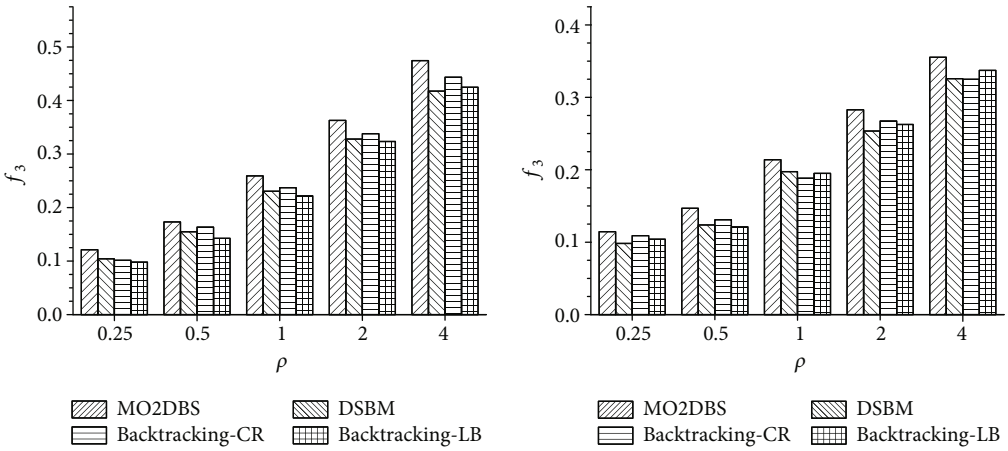
(a) Transmission delay obtained in NSFNET when $N_D = 4N_V/5$ (b) Transmission delay obtained in CHNNET when $N_D = 4N_V/5$

Figure 9: Transmission delay obtained when $N_D = 4N_V/5$.



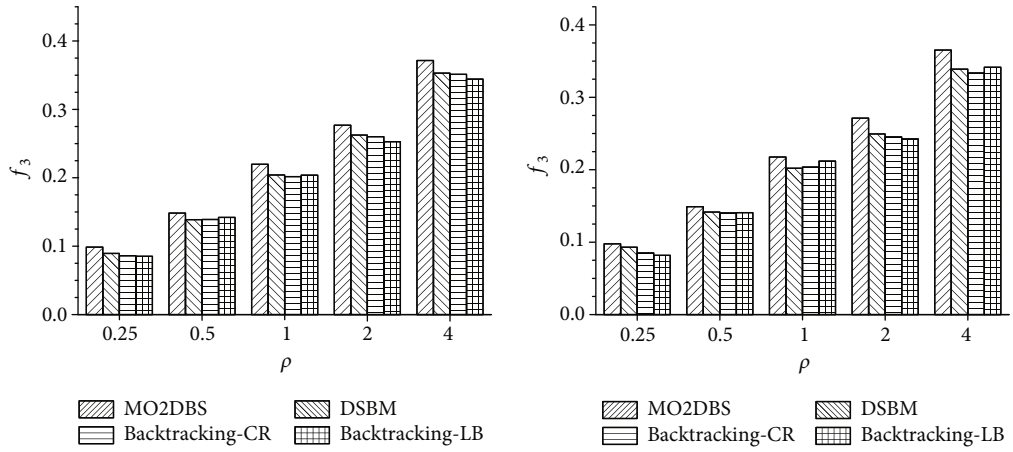
(a) Load degree obtained in NSFNET when $N_D = N_V/5$ (b) Load degree obtained in CHNNET when $N_D = N_V/5$

Figure 10: Load degree obtained when $N_D = N_V/5$.



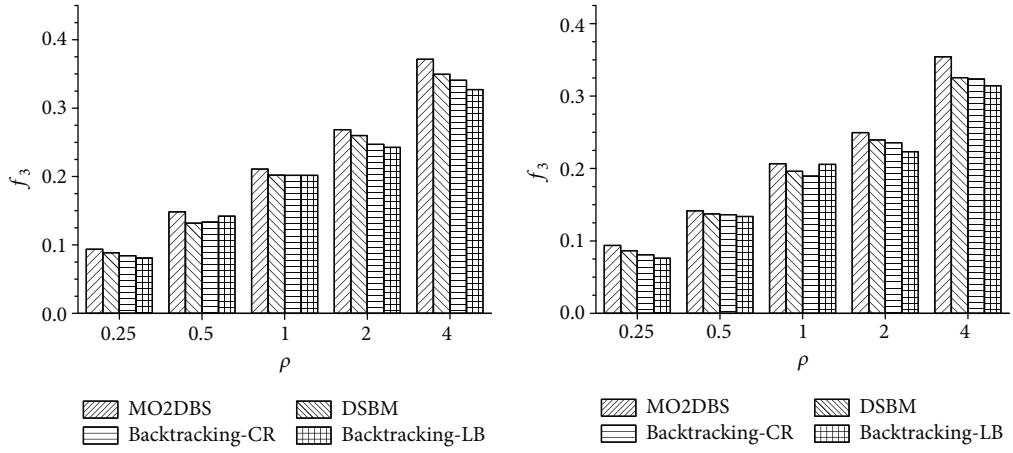
(a) Load degree obtained in NSFNET when $N_D = 2N_V/5$ (b) Load degree obtained in CHNNET when $N_D = 2N_V/5$

Figure 11: Load degree obtained when $N_D = 2N_V/5$.



(a) Load degree obtained in NSFNET when $N_D = 3N_V/5$ (b) Load degree obtained in CHNNET when $N_D = 3N_V/5$

Figure 12: Load degree obtained when $N_D = 3N_V/5$.



(a) Load degree obtained in NSFNET when $N_D = 4N_V/5$ (b) Load degree obtained in CHNNET when $N_D = 4N_V/5$

Figure 13: Load degree obtained when $N_D = 4N_V/5$.

compared to optimize the target scene, with the aim to optimize the path overhead scene only consider only logical link by mapping the underlying network path length, because the algorithm allows multiple services mapping to the same underlying node, therefore, the probability of a single underlying node load multiple services to improve, make the path length shorter, link resource consumption is reduced. The resources saved make it possible to build more service paths, so the success rate of service path building is slightly higher. However, since the former does not consider the load of nodes and links, the possibility of generating resource fragments increases and the probability of receiving service requests with large resource demands decreases. Therefore, the long-term average income of the two is relatively close.

When only a single optimization objective is considered, the algorithm achieves the best performance in the corresponding evaluation indexes, and when only the path cost and load intensity are considered, the two algorithms achieve close performance. Figure 5 shows the average income and the ratio of average cost. The index can reflect the resources used for the efficiency of the algorithm, only target path overhead. The algorithm achieved the lowest cost and highest earnings; therefore, revenue/cost ratio is highest, which can be the most efficient use of the resources, but only to delay as the goal, the highest cost, income minimum, resource utilization are the worst. According to the above comparison and analysis, if only a single optimization objective is considered when constructing the service path, it is difficult to achieve the mapping effect of low overhead, low latency, high income, and high reception rate. Therefore, the three factors need to be considered comprehensively. (1) Both of them are optimized to minimize end-to-end delay and only consider to avoid overuse of resources, without considering efficient resource allocation and load balancing. (2) Although the candidate graphs constructed by both of them are different (hierarchical graph and step search graph), Dijkstra shortest path algorithm is applied to construct the service path on the candidate graph, and the resources are reserved during the construction of the shortest path spanning tree. Due to the NP complexity of service path construction and the insufficiency of Dijkstra algorithm in dealing with resource reservation, effective service paths

have not been discovered and constructed in both of them, resulting in low success rate and low profit of construction. As in hierarchical graph, in order to minimize the time delay, Dijkstra algorithm in the map service does not take into account the service load node at this time, resource utilization, minimum value is always the option of vertical side, in the corresponding node at the same time to make the same request multiple services, resource consumption too fast, the formation of resource bottleneck, affect the subsequent service path to build. From the perspective of the algorithm, the reasons are as follows: (1) the algorithm utilizes the population evolution effect of the particles to expand the search scope and can effectively find the existing service path. (2) The algorithm comprehensively considers the two factors of service path cost and load intensity, reduces link resource consumption, balances network load, and can effectively improve revenue and build more service paths.

As the number of service requests increases and the amount of available resources decreases, the end-to-end delay increases gradually. As the resource bottleneck appears, the number of failed service path construction increases. When the curve flattens until convergence, it indicates that no new service path is constructed. Its long-term average revenue and cost-of-income ratio were significantly lower than those of algorithm, and its average cost and load intensity were significantly higher than those of the algorithm. The main reason is that the algorithm does not consider the path length and load of the underlying network when initializing and updating particle positions, resulting in high link resource consumption and many resource fragments.

4.4. Complexity of Proposed Algorithm. In the proposed algorithm, K shortest paths should be calculated for each connection request in advance, and its complexity is $o(K(N^s)^2)$. There are N_R connection requests, so the complexity for all connection requests for calculating K shortest paths is $o(K(N^s)^2 N_R)$. The fitness function calculation in the proposed algorithm remains the most complicated, and its complexity is $o(2G_m P_s (N^s)^2 N_F)$, where G_m, P_s, and N_F denote iteration times, population size, and maximum of frequency slots. Therefore, the complexity of proposed algorithm is $O(K(N^s)^2 N_R + 2G_m P_s (N^s)^2 N_F)$.

5. Conclusion

Network virtualization functions to the network layer and transport layer network function in the form of a software unit in the core network routers or server; using the controller to the function of network to carry on the arrangement and combination, to build an end-to-end service path, and to support a variety of custom service solves the present middle box deployment flexibility and scalability of the defect. Aiming at the key problem of service path construction in the above service delivery mechanism, this paper, from the perspective of reducing delay, reducing overhead, and balancing load, transforms multiple indexes into a single service path quality evaluation index by weighted sum method and establishes an integer linear programming model of service path construction problem. Then, a discrete particle swarm optimization model for service path construction was established according to the characteristics of particle swarm parallel search, and the proposed algorithm was designed for service path construction. In order to further reduce resource consumption, reduce the probability of resource fragmentation, and improve the convergence speed of particle swarm optimization, an optimization strategy was proposed to guide the initialization and updating of particle positions. Combined with the proposed algorithm, the proposed algorithm was obtained. Simulation results show that compared with the existing service path construction algorithms, the proposed algorithm can effectively optimize the comprehensive quality of service path and improve the success rate of service path construction and long-term average return. Compared with the rand-proposed algorithm, which uses random method to initialize and update particle positions, the proposed algorithm formulates evaluation criteria for candidate nodes and paths, which provides effective guidance for particle flight and further optimizes the performance of the algorithm.

Data Availability

All the data can be found in the paper.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the Doctoral Research Initiation Program of Weinan Normal University (Nos. 20RC15 and 20RC14) and the Third Group of Outstanding Talents Supporting Projects in Shaanxi Colleges and Universities (20RC03).

References

[1] R. Mijumbi, J. Serrat, J. L. Gorricho, N. Bouten, F. D. Turck, and R. Boutaba, "Network function virtualization: state-of-the-art and research challenges," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 236–262, 2016.

[2] Y. Li and M. Chen, "Software-defined network function virtualization: a survey," *IEEE Access*, vol. 3, pp. 2542–2553, 2017.

[3] H. Xuan, X. Zhao, Z. Liu, J. Fan, and Y. Li, "Energy efficiency opposition-based learning and brain storm optimization for vnf-sc deployment in iot," *Wireless Communications and Mobile Computing*, vol. 2021, Article ID 6651112, 9 pages, 2021.

[4] I. F. Akyildiz, S. C. Lin, and P. Wang, "Wireless software-defined networks (W-SDNs) and network function virtualization (NFV) for 5G cellular systems: an overview and qualitative evaluation," *Computer Networks*, vol. 93, pp. 66–79, 2015.

[5] M. Xia, M. Shirazipour, Y. Zhang, H. Green, and A. Takacs, "Optical service chaining for network function virtualization," *IEEE Communications Magazine*, vol. 53, no. 4, pp. 152–158, 2015.

[6] Y. Hui, Z. Jie, Y. Ji, T. Rui, and Y. Lee, "Performance evaluation of multi-stratum resources integration based on network function virtualization in software defined elastic data center optical interconnect," *Optics Express*, vol. 23, no. 24, p. 31192, 2015.

[7] H. Cao, Y. Zhu, Z. Gan, and L. Yang, "A novel optimal mapping algorithm with less computational complexity for virtual network embedding," *IEEE Transactions on Network & Service Management*, vol. 15, no. 1, pp. 356–371, 2018.

[8] C. Liang, F. R. Yu, and X. Zhang, "Information-centric network function virtualization over 5G mobile wireless networks," *IEEE Network*, vol. 29, no. 3, pp. 68–74, 2015.

[9] S. Sun, M. Kadoch, L. Gong, and B. Rong, "Integrating network function virtualization with SDR and SDN for 4G/5G networks," *IEEE Network*, vol. 29, no. 3, pp. 54–59, 2015.

[10] X. Fu, R. Yu, J. Wang, Q. Qi, and J. Liao, "Performance optimization for blockchain-enabled distributed network function virtualization management and orchestration," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 6, pp. 6670–6679, 2020.

[11] B. Chen, J. Zhang, W. Xie, J. P. Jue, Y. Zhao, and G. Shen, "Cost-effective survivable virtual optical network mapping in flexible bandwidth optical networks," *Journal of Lightwave Technology*, vol. 34, no. 10, pp. 2398–2412, 2016.

[12] Y. Yu, X. Bu, K. Yang, H. K. Nguyen, and Z. Han, "Network function virtualization resource allocation based on joint benders decomposition and ADMM," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 2, pp. 1706–1718, 2020.

[13] H. Hawilo, M. Jammal, and A. Shami, "Exploring microservices as the architecture of choice for network function virtualization platforms," *IEEE Network*, vol. 33, no. 2, pp. 202–210, 2019.

[14] J. Wu, M. Dong, K. Ota, J. Li, W. Yang, and M. Wang, "Fog-computing-enabled cognitive network function virtualization for an information-centric future internet," *IEEE Communications Magazine*, vol. 57, no. 7, pp. 48–54, 2019.

[15] M. Karimzadeh-Farshbafan, V. Shah-Mansouri, and D. Niyato, "A dynamic reliability-aware service placement for network function virtualization (NFV)," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 2, pp. 318–333, 2020.

[16] T. Lin and Z. Zhou, "Robust network function virtualization," *Networks*, vol. 75, no. 4, pp. 438–462, 2020.

[17] Y. Li, Y. Zhao, B. Li et al., "Joint balancing of it and spectrum resources for selecting virtualized network function in inter-datacenter elastic optical networks," *Optics Express*, vol. 27, no. 11, pp. 15116–15128, 2019.

[18] C. Yao, X. Wang, Z. Zheng, G. Sun, and L. Song, "EdgeFlow: open-source multi-layer data flow processing in edge computing for 5G and beyond," *IEEE Network*, vol. 33, no. 2, pp. 166–173, 2019.

[19] Z. Luo, C. Wu, Z. Li, and W. Zhou, "Scaling geo-distributed network function chains: a prediction and learning framework," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 8, pp. 1838–1850, 2019.

[20] D. M. Manias and A. Shami, "Making a case for federated learning in the internet of vehicles and intelligent transportation systems," *IEEE Network*, vol. 35, no. 3, pp. 88–94, 2021.

[21] L. Gu, D. Zeng, S. Tao et al., "Fairness-aware dynamic rate control and flow scheduling for network utility maximization in network service chain," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 5, pp. 1059–1071, 2019.

[22] S. Cheng, Q. Qin, J. Chen, and Y. Shi, "Brain storm optimization algorithm: a review," *Artificial Intelligence Review*, vol. 46, pp. 445–458, 2015.

[23] H. Duan, S. Li, and Y. Shi, "Predator–prey brain storm optimization for dc brushless motor," *IEEE Transactions on Magnetics*, vol. 49, no. 10, pp. 5336–5340, 2013.

[24] X. Xue and Y. Wang, "Optimizing ontology alignments through a memetic algorithm using both matchfmeasure and unanimous improvement ratio," *Artificial Intelligence*, vol. 223, pp. 65–81, 2015.

[25] J. Xue, Y. Wu, Y. Shi, and C. Shi, "Brain storm optimization algorithm for multi-objective optimization problems," *Lecture Notes in Computer Science*, vol. 7331, no. 4, pp. 513–519, 2012.

[26] M. El-Abd, "Global-best brain storm optimization algorithm," *Swarm and Evolutionary Computation*, vol. 37, pp. 5336–5340, 2017.

[27] A. Ss, A. Wt, B. Mr et al., "Network service chaining with optimized network function embedding supporting service decompositions," *Computer Networks*, vol. 93, pp. 492–505, 2015.

[28] A. Hammad, R. Nejabati, and D. Simeonidou, "Novel methods for virtual network composition," *Computer Networks*, vol. 67, no. jul. 4, pp. 14–25, 2014.