Journal of Computing and Electronic Information Management
ISSN: 2413-1660 | Vol. 12, No. 1, 2024

The Optimized Deployment of Service Function Chain Based on Reinforcement Learning

Yibo Zhang

Department of Computer Science, Hanyang University, Seoul, South Korea
zhangyibo@hanyang.ac.kr

Abstract: With the rapid development and application of technologies such as artificial intelligence, the Internet of Things, and cloud computing, data is showing explosive growth. In order to address the rising energy consumption due to the increasing number of devices in the traditional network architecture, software-defined networking and network function virtualization have been proposed. In this paper, we propose a reinforcement learning model based on actor-critic architecture. The service function chain deployment problem is mathematically modeled, and minimizing the total service function chain delay is taken as the optimization objective. The experimental results demonstrate that the service function chain deployment algorithm proposed in this paper is improved in terms of total system latency.

Keywords: Service Function Chain; Reinforcement learning; Network Function Virtualization.

1. Introduction

As technology increasingly enriches people's lives, people also put forward higher requirements for network services, requiring network architectures to provide higher bandwidth, more reliable and lower-latency services, and greater terminal access capabilities. However, the traditional middleware network structure based on specific network physical facilities can no longer meet the needs of network services. When higher requirements are put forward for network functions, it is necessary to expand network capacity by adding network equipment and continuously updating the network. At the same time, network requests and network traffic are dynamically changing, but traditional network physical hardware facilities are fixed and difficult to expand, which will greatly reduce network efficiency. In addition, these physical network facilities have dedicated suppliers, which will greatly increase the hardware capital expenditure and equipment management operation and maintenance expenditure of network operators and virtual network providers [1]. In order to solve the above problems, Service Function Chain (SFC) based on Network Function Virtualization (NFV) technology came into being [2]. Network Functions Virtualization is an innovative network architecture. It virtualizes traditional hardware network functions into virtual network function instances, connects a series of network functions into a service function chain, and deploys them in a certain order to run on general-purpose servers, which replaces traditional specific network function hardware facilities. These virtual network function instances can be deployed at any time or migrated to any available general-purpose server.

An efficient and reasonable service function chain deployment strategy can bring huge network performance benefits. Current research mainly focuses on optimizing latency [3], optimizing resource utilization [4], optimizing energy consumption, etc. The service function chain deployment problem is defined as an NP-HARD problem [5]. How to solve this problem in polynomial time is the core of the research. Literature [6] designed a dynamic programming algorithm, which can quickly and effectively deploy virtual network functions. The idea of the algorithm is to gradually split a large problem into multiple small problems for solution. Literature [7] proposed an R-ILP model based on multi-batch solution to simplify the integer linear programming problem, and proposed a combination of online and batch processing to process network requests. Literature [8] proposes a model that can solve the service function chain scheduling problem in polynomial time, and this model can reduce network energy consumption without destroying network performance. The above research uses heuristic algorithm to solve the problem, but when the problem constraints increase and the feasible solution space becomes smaller. The performance of the algorithm will decrease, and it is often limited to the local optimal solution. In view of the above problems, this paper proposes a service function chain deployment algorithm based on neural network optimization, which can improve performance.

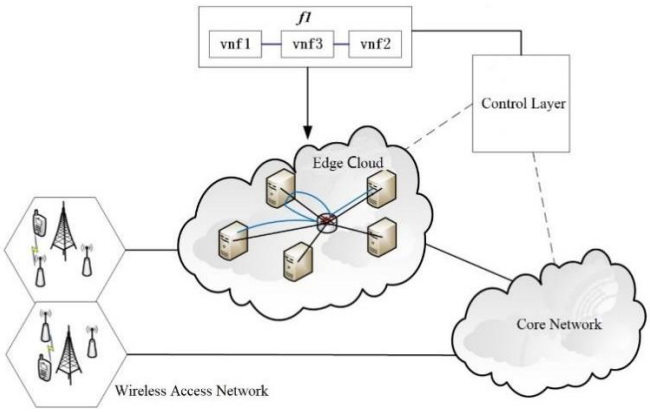

Figure 1. Mobile Edge Cloud Scenario

2. System Model and Problem Description

2.1. System model

The scenario considered in this chapter is the service function chain deployment problem of Mobile Edge Cloud, as shown in Figure 1. First, the physical network layer needs

to be defined. The physical layer network includes multiple general-purpose servers, which are used to process incoming data to satisfy the virtual network function and transmit the processed data to the next general-purpose server mapped by the virtual network function for further processing. Define the general server set as $Z = (z_1, z_2, z_3, \cdots, z_n)$, z_i represents a general server, and i is the index of the general server. Each server has its own specific resources (computing resources, cache resources, etc.), and r_{z_i} is defined to represent the total resources of a general server z_i. The topology considered in this chapter is a star topology, where each general-purpose server has a physical link to the controller. The data flow processed by the general server will be transmitted to the controller, and then the controller will control the data flow to the next general server for the next network service. Define the $L = (l_1, l_2, l_3, \cdots, l_n)$ as a set of physical links, l_i represents a physical link, and a general server corresponds to a physical link one by one, so the index of a physical link is the same as that of a general server. At the same time, each physical link has a specific bandwidth resource, and b_{l_i} is defined to represent the total bandwidth of the physical link l i. The network service requested by the user is processed by a service function chain, and a service function chain is composed of one or more virtual network functions. Define a service function chain set F, where f_x represents a service function chain in F, $f_x \in F$, and x is the index of the service function chain. Define Q_{f_x} as the set of virtual network functions f_m^{vnf} in the service function chain f x requested by the user, $f_m^{vnf} \in Q_{f_x}$, m is the index of the virtual network function in the service function chain, and the user's request is determined by various virtual network functions on the service function chain Network functions are processed in an orderly manner.

2.2. Optimization problems

In order to handle a large number of network requests, if operators just blindly increase general-purpose servers and physical links, the resource usage of general-purpose servers and physical links will be greatly reduced, and the total power will increase rapidly with the increase of requests. Therefore, it is particularly important to provide a more efficient, energy-saving, and intelligent service function chain deployment strategy for general-purpose servers deployed in cloud nodes, edge nodes, or data centers. Therefore, this paper constructs an optimization problem: taking the delay requirements of user terminal requests, general server resources and physical link resources as constraints, to minimize the total power of the service function chain, where the total power is divided into general server power and total physical link power. On the premise of ensuring the deployment quality of the service function chain, the efficiency of network resource usage is improved.

First, the total power of general servers is divided into idle power and working power. Define a logical variable v_{z_i} to represent whether the server z_i is in the working state, and the general server is deployed with a virtual network function instance, which is the working state ($v_{z_i} = 1$). Define the current resource usage L_{z_i} of the general server as the total usage of computing resources of the virtual network function instances deployed on the general server z_i. The power and resource usage of a general server are positively correlated when it is working, that is, the higher the current resource usage of the general server, the higher the power of the general server. The current resource usage L_{z_i} of the general server and the total power W^Z of the general server are calculated as follows:

$$L_{z_i} = \sum_{f_x \in F} \sum_{f_m^{vnf} \in Q_{f_x}} r_{f_m^{vnf}}^{demand} \times k_{z_i}^{f_m^{vnf}} \quad (1)$$

$$W^Z = \sum_{z_i \in Z} [v_{z_i}] \times \left(w_{z_i}^{work} - W_{z_i}^{idle}\right) \times \frac{L_{z_i}}{r_{z_i}} + W_{z_i}^{idle} \times (1 - v_{z_i}) \quad (2)$$

Among them, $w_{z_i}^{work}$ represents the full load power of the general server z_i, $r_{f_m^{vnf}}^{demand}$ means the amount of resources required by the virtual network function f_m^{vnf}, $k_{z_i}^{f_m^{vnf}}$ is a logical variable indicating whether the virtual network function f_m^{vnf} is deployed on the general server z_i, $W_{z_i}^{idle}$ suggests the idle power of the general server z_i .

The second part is the total physical link power W^L , which is defined as the total power of the physical link transmission service function chain data flow. If the bandwidth requirement of the virtual network function is greater than the available bandwidth of the current link, the virtual network function cannot be deployed on the general server connected to this link. Define the current resource usage L_{l_i} of the physical link as the total bandwidth resource demand of the service function chain deployed on the link. The physical link power is positively correlated with the current resource usage of the physical link, that is, the higher the bandwidth resource usage of the physical link, the higher the power of the physical link. The bandwidth L_{l_i} used by the physical link and the total power W^L of the physical link are calculated as follows:

$$L_{l_i} = \sum_{f_x \in F} \sum_{f_m^{vnf} \in Q_{f_x}} b_{f_m^{vnf}}^{f_x} \times k_{z_i}^{f_m^{vnf}} \quad (3)$$

$$W^L = \sum_{l_i \in L} w_L^{band} \times L_{l_i} \quad (4)$$

where w_L^{band} represents the physical link power per unit bandwidth, and $b_{f_m^{vnf}}^{f_x}$ means the bandwidth required by the virtual network function f_m^{vnf} in the service function chain f_x.

Therefore, the total power W^F of the deployed service function chain can be defined as the sum of the total power of the general server and the total power of the physical link. The problem modeling is specifically expressed as follows:

$$\min_{k_{z_i}^{f_m^{vnf}}, v_{z_i}} W^F = \min_{k_{z_i}^{f_m^{vnf}}, v_{z_i}} (W^Z + W^L) \quad (5)$$

s.t.

$$C1: \sum_{z_i \in Z} k_{z_i}^{f_m^{vnf}} = 1, \forall f_m^{vnf} \in Q_{f_x} \quad (6)$$

$$C2: L_{z_i} \le r_{z_i}, \forall z_i \in Z \quad (7)$$

$$C3: L_{l_i} \le b_{l_i}, \forall l_i \in L \quad (8)$$

$$C4: \sum_{z_i \in Z} \sum_{f_m^{vnf} \in Q_{f_x}} d_{f_m^{vnf}}^{com} \times k_{z_i}^{f_m^{vnf}} + \sum_{l_i \in L} \sum_{f_m^{vnf} \in Q_{f_x}} d_{l_i}^{tran} \times k_{z_i}^{f_m^{vnf}} \le d^{f_x}, \forall f_x \in F \quad (9)$$

Among them, b_{l_i} represents the total bandwidth of the physical link l_i, d^{f_x} indicates the user's maximum tolerable delay of the service function chain f_x, $d_{f_m^{vnf}}^{com}$ means the processing delay of the virtual network function f_m^{vnf}, and $d_{l_i}^{tran}$ suggests the transmission delay of the physical link l_i. Constraint C1 indicates that any one virtual network function on service function chain f_x can only be deployed on one of the generic servers sets. Constraint C2 ensures that the total

resource requirements of all virtual network functions deployed on a common server cannot exceed the total resources of the common server. Constraint C3 makes sure that the total bandwidth requirements of all virtual network functions deployed on a common server is not more than the total bandwidth of the links connected to the common server. Constraint C4 enforces that the sum of processing latency and transmission latency of any one service function chain cannot exceed the maximum tolerable user latency of that service function chain.

3. Algorithm Description

When performing service function chain deployment, network requests are random. Therefore, the resource requirements for generic servers, physical links are dynamically changing, which leads to the action space dimension becoming huge and without a priori knowledge. At the same time, if the neural combinatorial optimization model is used directly, some internal constraints of the service function chain deployment problem are not considered, resulting in deployment results that do not satisfy the requirements. To solve the above problems, this paper proposes the Neural Combinatorial Optimization Service Function Chain Deployment Algorithm (NCOSFC) based on neural combinatorial optimization. The algorithm combines neural combinatorial optimization with constrained optimization problem and incorporates reinforcement learning between the intelligent body and the deployment environment to obtain the constraints in the deployment environment. The NCOSFC algorithm framework is shown in Figure 2, and for the convenience of illustration algorithm flow, the deployment vector $p^{f_x} = (p_1, p_2, p_3, \cdots, p_m), p_i \epsilon Z$ is defined.

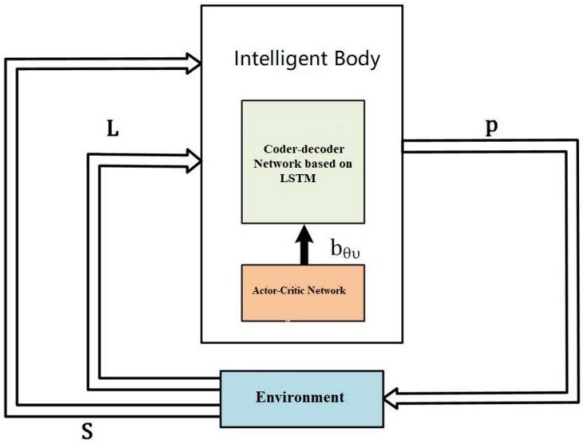

Figure 2. NCOSFC algorithm architecture diagram

3.1. Network model

The network model of the NCOSFC algorithm is divided into an actor network and a critic network based on an LSTM encoder-decoder. The actor network uses the method of policy gradient to train the deployment strategy $\pi_\theta(p^{f_x}|f_x)$, the purpose is to increase the probability of deployment strategies with low total deployment power of the service function chain. To simplify the description, replace p^{f_x} with p. First, an objective function is defined to determine the effect of the deployment strategy. In this problem, the objective function is defined as the requested service function chain deployment power expectation, and the formula is as follows:

$$J_E^\pi(\theta) = E_{f_x \in F}[E_{p \in \pi_\theta[\cdot|f_x]}[W^F(p)]] \quad (10)$$

In order to satisfy the constraints, a constraint on the desired reward is required. Define dissatisfaction J_C^π as the accumulation of three metrics, namely, generic server resource exceedance, physical link bandwidth exceedance and service function chain delay exceedance:

$$J_C^\pi(\theta) = E_{f_x \in F}[J_C^\pi(\theta|f_x)] \quad (11)$$

At this point, the optimization problem is transformed into finding a deployment strategy that minimizes the expectation of the total power of the service function chain. Constraining with dissatisfaction absorbs the constraints into the objective function, making the problem much less difficult to solve. The optimization of the objective function obtains:

$$\min_\theta J_L^\pi(\theta) = \min_\theta [J_E^\pi(\theta) + \sum_i \lambda_i \times J_{C_i}^\pi(\theta)] \quad (12)$$

where $J_L^\pi(\theta)$ is the Lagrangian objective function and λ_i is the Lagrangian multiplier or penalty factor.

Monte Carlo sampling is used to approximate the gradient representation and stochastic gradient descent is used to calculate the training weight parameter θ. For the objective function gradient solution, the log-likelihood method is used to approximate the representation:

$$\theta_{i+1} = \theta_i + \beta \nabla J_L^\pi(\theta) \quad (13)$$

$$L(p|f_x) = W^F(p|f_x) + \sum_i \lambda_i \times C_i(p|f_x)] \quad (14)$$

$$\nabla J_L^\pi(\theta) \approx \frac{1}{B}\sum_{j=1}^{B}(L(p_j|f_j) - b_{\theta_v}(f_j)) \times \nabla_\theta log \pi_\theta(p_j|f_j) \quad (15)$$

Among them, $W^F(p|f_x)$ represents the deployment total power function of the service function chain f, $C_i(p|f_x)$ means the dissatisfaction function, $L(p|f_x)$ suggests the constrained total power function, and $b_{\theta_v}(f_j)$ is the baseline function. This subsection also presents a deep neural network model based on the encoder-decoder for the training of policy gradients. The encoder adopts a Long Short-Term Memory (LSTM) network, and the decoder applies the same LSTM network of equal dimensionality. The encoder takes advantage of the memorability of the LSTM to efficiently extract the characteristics of the virtual network function sequence and input the context vector C into the decoder network. The decoder outputs the deployment vectors of the virtual network functions through the LSTM network. The network model is a Sequence-to-Sequence model based on the Long Short-Term Memory (LSTM) architecture, as shown in Figure 3.

In addition to make the model converge quickly in the policy network, a baseline function $b_{\theta_v}(f)$ is used to reduce the variance of the policy gradient. The baseline function in this problem is defined as the predicted value of the total constraint power $L(p_j|f_j)$ for deploying this service function chain, and a neural network is used to train it. The input of this neural network is the sequence of service function chains requested to be deployed and the output is the prediction of the total constrained power to deploy this service function chain $b_{\theta_v}(f_j)$. During the exploration of the intelligence, the baseline function guides the intelligence to follow the predicted total constrained power of the neural network model for the exploration training. This neural network model considers the mean square error between the baseline function and the true constrained total power of the deployment policy as the loss function, and the objective function $L(\theta_v)$ is calculated as follows:

$$L(\theta_v) = \frac{1}{B}\sum_{j=1}^{B}\left\|b_{\theta_v}(f_j) - L(p_j|f_j)\right\|^2 \quad (16)$$

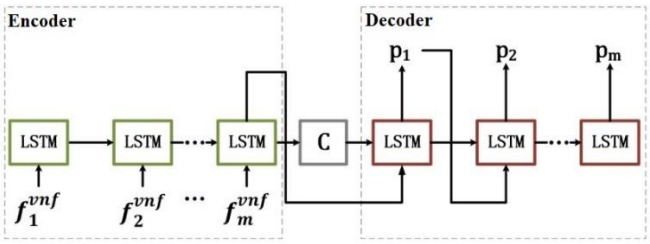

Figure 3. Actor network model for encoder-decoder based LSTM

3.2. Algorithm implementation flow

The NCOSFC algorithm is trained with an encoder LSTM network that learns the features of the input sequence and aggregates them into a context vector. The critic network predicts the constrained total power of the requested service function chain deployment policy, which is also the baseline function $b_{\theta_v}(f)$. The critic network evaluates this deployment strategy after the actor has made a deployment strategy. The probability of taking this strategy is increased or decreased depending on the total power of the service function chain deployment strategy.

In addition, the selection of the learning rate of the network model affects the training effect. If the learning rate is set too large, it will cause the network model to fluctuate on both sides of the optimal solution, resulting in over-fitting or under-fitting. If the learning rate is set too small, the training speed of the network model will be greatly reduced, increasing the training time, and wasting computational resources. Therefore, an appropriate learning rate is particularly important. In this paper, the particle swarm algorithm is used to select the optimal learning rate for the NCOSFC network model. The particle swarm algorithm is a classical stochastic search algorithm, which is often applied in optimization problems in various fields. The equation of particle update rate and position is as follows:

$$v_{i+1} = w \times v_i + c_1 \times r_1 \times (P_i - x_i) + c_2 \times r_2 \times (G - x_i) \quad (17)$$

$$x_{i+1} = x_i + v_i \quad (18)$$

Algorithm 1: Particle Swarm Optimization for NCOSFC

Input: Number of iterations T, number of particles N

Output: G is the NCOSFC optimal learning rate

1: Initialize the position, velocity, individual extremum and global optimal value of all particles

2: for i = 0 → T do

3: for i = 0 → N do

4: Calculate the fitness of particle i at the current position x_i according to the objective function

5: Compare this fitness with P_i, if it is smaller than P_i, update P_i to this fitness

6: end for

7: Compare the individual extremum of all particles, and set the smallest individual extremum as the global optimal value G

8: for i = 0 → T do

9: Update the velocity v_i of particle i according to formula (17)

10: Update the position x_i of particle i according to formula (18)

11: end for

12: end for

Where x_i denotes the position of particle i, v_i represents the velocity of particle i; c_1 and c_2 mean the learning factors in the particle swarm algorithm; ω suggests the inertia factor in the particle swarm algorithm to regulate the global and local search ability of the algorithm, with larger ω denoting stronger global search ability and smaller ω denoting stronger local search ability; P_i and G indicate the individual extreme value of particle i and the global optimal value among all particles, respectively. Two random floating point numbers r 1 and r 2 are introduced to increase the random search capability of the algorithm. In this section, the particle swarm algorithm is used to optimize the learning rate of the NCOSFC model to achieve model optimization, where the objective function is set to the constrained total power function $L(p|f_x)$ of the NCOSFC model, and the global optimum is the optimal learning rate of the NCOSFC model. The particle swarm algorithm flow is shown in Algorithm 1.

4. Experimental Results and Analysis

To simplify the physical network model, a star network topology was used for the experiment structure, as shown in Figure 4. In the figure, the physical network structure is deployed with 5 general-purpose servers and 5 physical links, and the total resources of each general-purpose server are r_z and the total bandwidth resources of each physical link are b_l. For the parameters related to the service function chain, the maximum tolerated delay of the service function chain is set to 60ms. In this experiment, the number of service function chains is taken as an independent variable, and 1, 5, 10, 15, 20, 25, 30, 35, 40 are respectively taken to compare the performance of the service function chain deployment algorithm. The comparison methods adopted are the First Fit Algorithm [9] and the Gecode Algorithm [10] respectively.

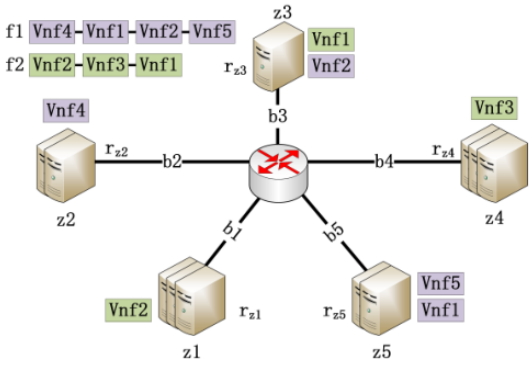

Figure 4. Network topology diagram

The total system latency of each algorithm in this experiment is observed by varying the number of service function chains. The total system latency is defined as the sum of the service function chain processing latency and the transmission latency. As shown in Figure 5, the total system latency of the First Fit Algorithm is similar to the NCOSFC algorithm and the Gecode Algorithm when there are few service function chain requests. As the number of service function chain requests increases, the available resources of the generic server and physical links become less, and the performance of the First Fit Algorithm weakens. However, the total system latency of the NCOSFC algorithm is still similar to the Gecode Algorithm. The particle swarm optimized NCOSFC model achieves better results when deploying more than 20 service function chains.

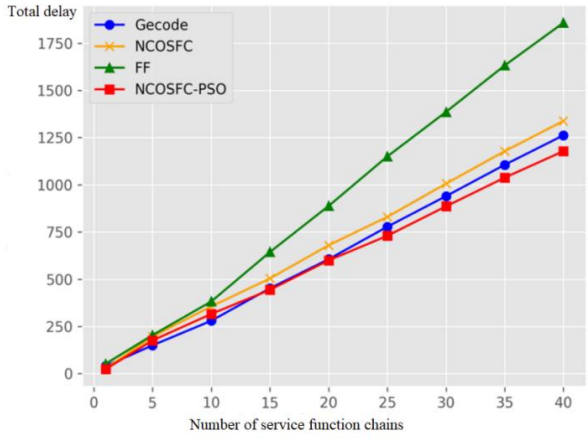

Figure 5. Total system latency for different number of service function chain requests

5. Conclusions

This paper mathematically models the service function chain deployment problem by minimizing the total service function chain power as the optimization objective and considering the service function chain latency, common server resources and physical link resources as the constraints. In this paper, the solving problem is researched using a reinforcement learning model based on the actor-critic architecture. Finally, the NCOSFC algorithm is compared with the First Fit Algorithm and the Gecode Algorithm for different number of service function chains. The experimental results demonstrate that the service function chain deployment algorithm studied in this paper outperforms the comparison algorithm.

References

[1] Sherry J, Hasan S, Scott C, et al. Making middleboxes someone else's problem: Network processing as a cloud service[J]. ACM SIGCOMM Computer Communication Review, 2012, 42(4): 13-24.

[2] Mijumbi R, Serrat J, Gorricho J L, et al. Network function virtualization: State-of-the-art and research challenges[J]. IEEE Communications surveys & tutorials, 2015, 18(1): 236-262.

[3] Németh B, Molner N, Martín-Pérez J, et al. Delay and reliability-constrained VNF placement on mobile and volatile 5G infrastructure[J]. IEEE Transactions on Mobile Computing, 2021, 21(9): 3150-3162.

[4] Hejja K, Hesselbach X. Offline and online power aware resource allocation algorithms with migration and delay constraints[J]. Computer networks, 2019, 158: 17-34.

[5] Addis B, Belabed D, Bouet M, et al. Virtual network functions placement and routing optimization[C]//2015 IEEE 4th International Conference on Cloud Networking (CloudNet). IEEE, 2015: 171-177.

[6] Ghribi C, Mechtri M, Zeghlache D. A dynamic programming algorithm for joint VNF placement and chaining[C]//Proceedings of the 2016 ACM Workshop on Cloud-Assisted Networking. 2016: 19-24.

[7] Soualah O, Mechtri M, Ghribi C, et al. Online and batch algorithms for VNFs placement and chaining[J]. Computer Networks, 2019, 158: 98-113.

[8] Shen G, Li Q, Jiang Y, et al. A four-stage adaptive scheduling scheme for service function chain in NFV[J]. Computer Networks, 2020, 175: 107259.

[9] Kumaraswamy S, Nair M K. Bin packing algorithms for virtual machine placement in cloud computing: a review[J]. International Journal of Electrical and Computer Engineering, 2019, 9(1): 512.

[10] Addis B, Belabed D, Bouet M, et al. Virtual network functions placement and routing optimization[C]//2015 IEEE 4th International Conference on Cloud Networking (CloudNet). IEEE, 2015: 171-177.