A Reliable Service Function Chain Orchestration Method Based on Federated Reinforcement Learning

Zhiwen Xiao(✉), Tao Tao, Zhuo Chen, Meng Yang, Jing Shang, Zhihui Wu, and Zhiwei Guo

China Mobile Information Technology Center, Beijing 100033, China
xiaozhiwen@chinamobile.com
http://it.10086.cn

Abstract. The novel cloud-edge collaborative computing architecture can provide more efficient and intelligent services close to users. Reliable service function chain orchestration among datacenters is critical to ensuring computing efficiency. In this study, a service orchestration model is proposed to improve the reliability while reducing cost. The solution is a federated reinforcement learning framework that shares decision-making experiences to obtain reliable and effective service orchestration results between different datacenter environments. The simulation results demonstrate that the proposed orchestration method reaches convergence faster and has a significant performance in terms of improving service reliability.

Keywords: Service function chain · Cloud-edge collaborative computing · Federated reinforcement learning · Reliability

1 Introduction

With the rapid development of Internet of Things and its corresponding online applications, analyzing large volumes of geographically distributed data is a critical issue for data analysts and real-time application decision-making. To tackle this issue, the computing mode has gradually changed from cloud-centric to cloud-edge collaboration to cater to real-time data processing requirements. Cloud service providers deploy multiple edge datacenters in multiple locations [1]. The key to provide stable computing services is that the service function chain formed by geo-distributed datacenters is reliable. Subsequently, network function virtualization (NFV) is proposed to virtualize the large-scale data tasks as VNF instances that can run on geo-distributed datacenters to provide users with quick access to their services [2].

Meanwhile, the orchestration based on NFV brings additional reliability problems [3,4]. Because it is necessary to introduce more complex functions on the basis of traditional physical nodes to support node virtualization. However,

H. Gao et al. (Eds.): CollaborateCom 2022, LNICST 460, pp. 173–189, 2022.
https://doi.org/10.1007/978-3-031-24383-7_10

due to the objective existence of software defects and other factors, virtual nodes built on reliable physical nodes may themselves be faulty. Existing research usually solves the reliability problem through a backup strategy [5] and network protection strategy [3], but it will bring additional and redundant equipment overhead. Considering reliability at the beginning of service deployment is a better solution. Therefore, service orchestration not only needs to consider the minimization of the orchestration overhead but also the reliability of the service. This paper establishes the orchestration model to jointly optimize the cost and service reliability.

To solve optimization and obtain service function chain orchestration decision, the reinforcement learning method is widely used [6–8]. Because of its dynamic decision-making adaptability, it can contribute to the reliability of the service function chain. However, the reinforcement learning model requires intensive interaction with network environment to learn an effective strategy. The reinforcement learning model in a single datacenter is trained based on the local data to learn local experience. The independent model only maintains good performance in the specific service and network environment, and cannot adapt to the environment of multiple datacenters. For geo-distribution datacenters in different regions, their user groups and user services may slightly overlap. At the same time, due to different services, there are also differences in the feature spaces of the datasets between the two datacenters [10,11]. Besides, the business data of different datacenters may be under the jurisdiction of different companies, and it is difficult to collect all business data to train the overall model. Therefore, service orchestration across datacenters cannot be limited to local data only. The non-interoperable datacenters are hard to share decision-making experiences leads to the reinforcement learning model must collect new training samples and retrain the neural network so that the model can converge when building a service chain in a new datacenter [9]. This process leads to slow convergence and poor model performance of the reinforcement learning, making it difficult to ensure the effect of service orchestration for geo-distributed datacenters.

It therefore remains necessary to address the aforementioned challenges to improve the convergence efficiency and stability of reinforcement learning model for reliable service orchestration in geo-distributed datacenters. Federated learning [12] provides a framework for collaborative orchestration for datacenters to solve the problem of non-sharing of experience between independent models. Federated learning aggregates the orchestration experience of multiple data centers, and delivers the aggregated model to each data center. By integrating the training parameters of reinforcement learning in each datacenter environment, optimal service orchestration decisions can be obtained.

The main contribution of this paper is as follows: We propose a service function chain orchestration method to jointly optimize cost and the service reliability for geo-distributed datacenters in the cloud-edge collaborative computing architecture. First, for the reliability assurance of service function chain orchestration in multiple datacenters, we model service function chain orchestration

problem with both cost function and reliability assessment of service function chain. Then, for cross-regional multi-datacenter scenarios, we introduce a federated reinforcement learning framework to obtain the orchestration decision. This method uses the reinforcement learning model as the training basis and then obtains the federated reinforcement model by fusing the training parameters in different datacenters. The model implemented in the training and new datacenter environments can achieve convergence faster and obtain the optimal decision with better service reliability while reducing cost.

The remainder of this paper is organized as follows: Sect. 2 provides an overview of related work into the service function chain orchestration. Section 3 outlines the federated reinforcement learning model for computing datacenters orchestration. Section 4 describes the simulation experiments and analyzes the performance of the proposed model. In the final section, we provide conclusions and recommendations for further research.

2 Related Work

In terms of service function chain orchestration, existing studies use different methods to solve this problem. Dieye et al. [13] modelled service function chain orchestration as an integer linear programming problem and proposed a cost-effective active VNF placement and linking algorithm. The algorithm can find the optimal number of VNFs and their positions to minimise costs while satisfying QoS. Sang et al. [14] constructed an integer linear programming model to solve the VNF dynamic placement problem. Yang et al. [15] used a path-based integer linear programming model to minimise network energy consumption when solving service orchestration problems. Kar et al. [16] designed a dynamic energy-saving model with M/M/c queuing network and minimum capacity strategy, improving machine utilisation and avoiding frequent changes in machine state. For the placement of service function chains, the authors defined an energy cost optimisation problem constrained by capacity and proposed a heuristic dynamic VNF chain placement solution. Varasteh et al. [17] proposed a fast heuristic framework that can effectively solve the power-aware and delay-constrained VNF placement and routing problems. Troia et al. [6] studied the application of reinforcement learning to perform dynamic SFC quotas in NFV-SDN-enabled metro core optical networks. The authors constructed a reinforcement learning system that optimises SFC quotas in multi-layer networks. Quan et al. [18] applied deep reinforcement learning (DRL) to solve the placement problem of virtual network function-forwarding graphs and developed a simulation platform based on the Mininet and containers to demonstrate the advantages of DRL over existing methods. Pei et al. [19] proposed a VNF placement algorithm (DDQN-VNFPA) based on a double deep Q-network using deep reinforcement learning technology. DDQN obtains the best solution from a considerable solution space, and DDQN-VNFPA places or releases VNF instances (VNFIs) according to threshold-based policies. This algorithm can improve overall network performance.

All researches above, which face single-agent service orchestration schemes, propose various optimization goals. However, when the environment changes and

training data is limited, performing high-quality resource orchestration decisions is difficult. Multi-agent service orchestration schemes have also been widely discussed. Shah et al. [7] pointed out that the service function chain orchestration problem under the constraints of IoT systems can be expressed as a Markov decision process (MDP). A multi-agent deep reinforcement learning algorithm can solve the MDP problem, where each agent serves a service function chain. They proposed two Q-networks in the specific implementation. One Q-network solves the service function chain placement problem. It generates virtual agent interactions with the environment to receive accumulated rewards and uses the learned experience to update the policy. The other updates the Q-value by tracking long-term policy change weight. Liu et al. [8] developed a multi-agent reinforcement learning framework that uses an independent learner-based multi-agent Q-learning (IL-based MA-Q) algorithm to solve distributed computing offloading problems in Edge Computing. However, none of these methods considers data security and interfaces protection issues between multiple edge networks.

Federated learning provides a reasonable framework for non-interoperable data in multi-agent collaborative computing. Huang et al. [10] proposed an extensible orchestration method based on federated reinforcement learning, which introduces federated learning into global model training and deep reinforcement learning into local model training to achieve an extensible services function chain. This method works as follows: First, this method divides the entire network into regions and assigns an agent in each region to train a local model of service function chain orchestration. Then, the cloud specifies an initial pre-trained model and sends it to each edge datacenter. Afterwards, each agent trains its local model and reports it to the cloud for global model aggregation through federated learning. Finally, it places the VNF into the network according to the learned policy. However, these federated learning training environments are different regions of the same network. This work is hard to apply in different datacenter environments or a new edge environment under the cloud-edge collaboration mode.

3 Proposed Method

In this section, the design proposal for the service function chain orchestration model is presented. We also describe our federated reinforcement learning framework to show the principle of the reliable computing service orchestration method.

3.1 Service Function Chain Orchestration Model

In order to construct the orchestration of service function chains, we model an orchestration problem. The following definitions are given to describe the orchestration problem of service function chains.

Definition 1. *The physical network $G_p = (V_p, E_p)$ is composed of a set of physical nodes V_p and a set of physical links E_P, with physical nodes $v \in V_p$ and physical links $e \in E_p$. A physical node represents a physical server carrying virtual functions, and a physical link is an actual link between physical nodes.*

Definition 2. *The virtual network $G_v = (V_v, E_v)$ comprises a virtual node set V_v and virtual link set E_v, with virtual node $f \in V_v$ and virtual link $z \in E_v$. The virtual node represents a VNF on the service function chain, and the virtual link represents the logical concatenation relationship between the VNFs.*

Definition 3. *For service $s \in S$, there is resource mapping $g_s = (g_s^V, g_s^E)$. Where S represents the service set, g_s^V represents the mapping $V_v \rightarrow V_p$ of virtual node set V_v to physical node set V_p, g_s^E represents the mapping $E_v \rightarrow E_p$ of virtual link set E_v to physical link set E_p.*

Definition 4. *$a_{f \rightarrow v}$ represents the act of placing virtual node f to physical node v, $a_{f \rightarrow v} \in g_s^V$. Similarly, $a_{z \rightarrow e}$ represents the act of placing virtual link z to physical link e, $a_{z \rightarrow e} \in g_s^E$.*

Definition 5. *$\lambda : \{\lambda_v^f, \lambda_e^z\}$ represents the VNF placement decision. λ_v^f means to execute a decision of $a_{f \rightarrow v}$. λ_e^z means to execute a decision of $a_{z \rightarrow e}$ The value range of λ is $\{0, 1\}$, 1 means to place on a physical node or link, and 0 means not to place. For $\forall \lambda \in \Pi^s$, Π^s represents a set of policies that map all virtual links on service s to physical links.*

Under the above definition, the service function chain orchestration problem can be expressed as follows: the service in the virtual network needs to find an optimal placement strategy to realize the one-to-one mapping of virtual resources to physical resources. The optimization goal is improving reliability and reducing operating costs. We define the calculation methods of cost and reliability as follows, and give a formalized optimization problem.

$$Cost = \sum_{x} \sum_{f} \lambda_v^f \cdot k_x + \sum_{z} \sum_{x,y} \lambda_e^z \cdot k_z \tag{1}$$

where k_x represents the cost of the unit resource on the node x, and k_z represents the cost of the unit resource on the link z.

The reliability of the service defined as r_s:

$$r_s = \prod_{e} r_e \cdot \prod_{v} r_v \tag{2}$$

where r_e represents the reliability of the physical link e and r_v represents the reliability of the physical node v. The reliability assessment process of the above physical links and nodes (r_e and r_v) can refer to [20].

Based on the above analysis, service function chain orchestration problem can be formulated as:

$$\begin{aligned} &\max_{\lambda} \frac{\beta r_s}{Cost} \\ s.t. \ &\sum_{z} \lambda_e^z \leq C_E(t) \\ &\sum_{f} \lambda_v^f \leq C_V(t) \\ &\sum\nolimits_{v \in V_p} a_{f \rightarrow v} = 1 \\ &\sum\nolimits_{e \in E_p} a_{z \rightarrow e} = 1 \end{aligned} \tag{3}$$

where β is an adjustment coefficient which controls the weight of the reliability and cost. The first constraint $\sum_{z} \lambda_e^z \leq C_E(t)$ is the resource constraint on links, $C_E(t)$ represents the total resources of physical link e at time t. The second constraint $\sum_{f} \lambda_v^f \leq C_V(t)$ is the resource constraint on nodes, $C_V(t)$ represents the total resources of the physical node v at time t. The third and fourth are constraints on decision variables. In order to simplify the analysis of the orchestration problem, this paper does not consider the backup of resources. At this time, $\sum_{v \in V_p} a_{f \to v} = 1$ means a virtual node can only be mapped to one physical node; meanwhile, $\sum_{e \in E_p} a_{z \to e} = 1$ means a virtual link can only be mapped to one physical link.

After modelling this problem, we build the federated reinforcement learning solution with the optimization goal of improving reliability and reducing operating costs in the next two sections.

3.2 Reinforcement Learning Model in Single Datacenter

The service function chain orchestration process can be split into the sequential placement of VNFs, accompanied by the connection of links. The impact of each VNF placement on the overall service function chain is related to the VNF placed previously. The reinforcement learning model can obtain the optimal policy by calculating the reward of each action, which is suitable for solving the step-by-step orchestration problem of the service function chain.

In order to solve the service function orchestration problem using the reinforcement learning model, several essential parts of the reinforcement learning model need to be defined and analyzed: state, action, reward and target. The target of service orchestration in this study is to improve service reliability. When designing the reward function, the reliability and cost factors of the service need to be considered.

State: $S_t = \{C(t), F_{new}, F_{old}\}$ represents the state at time t, where $C(t)$ represents the occupation of physical resources at time t, F_{new} represents the VNF node to be placed, and F_{old} represents the previously placed VNF node.

Action: $A_t = \{\lambda_t\}$represents the action at time t, where λ_t represents the placement decision at time t, and λ is defined in Definition 5.

Reward Function: Since the optimization objectives of orchestration are to improve the reliability of services and reduce costs, the reward function is defined by reliability and cost-benefit under resource-constrained. Then, the reward function R is presented by Eq. 4:

$$R = \begin{cases} \beta \cdot \frac{r_s \times 100}{Cost}, & if \sum_{z} \lambda_e^z \leq C_E(t) \ or \sum_{f} \lambda_v^f \leq C_V(t) \\ -1, & otherwise \end{cases} \tag{4}$$

The objective of reinforcement learning is to find the optimal policy $\pi^*(\mathrm{A}_t|O_t)$, which can obtain the maximum the reward from the initial state

O_t, as shown in Eq. 5:

$$\max_{\pi} E[\sum_{t=0}^{H} \gamma^t R(S_t, A_t, S_{t+1})|\pi] \tag{5}$$

In this work, we introduce Q-learning as the basis learner. Because Q-learning is a model-free reinforcement learning method that learns how to find the optimal action selection policy through interaction with the environment. An optimal policy can be found by updating the Q-table, which is the mapping table between the state-action and the estimated future reward. The update process of Q-table $Q(s_t, a_t)$ is presented by Eq. 6:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_{t+1} + \gamma Q(s_{t+1}, a') - Q(s_t, a_t)) \tag{6}$$

where $a \in \{0, 1\}$ is the learning factor, $\gamma \in \{0, 1\}$ is the discount factor and a' represents the behavior under strategy π. Then the currently selected action is presented by Eq. 7:

$$\pi(s_{t+1}) = \arg\max_{a'} Q(s_{t+1}, a') \tag{7}$$

In order to ensure the generalization function of Q-learning and avoid falling into the local optimum of the result, we generally use the ϵ-greedy method for action selection. The mechanism selects an optional action uniformly and randomly with a small probability ϵ of exploring and selects the current best action according to the above formula with a probability of $1 - \epsilon$.

The reinforcement learning model can obtain the optimal orchestration strategy after training. The limitation of this method is that it relies on a large amount of training experience, and the orchestration scheme cannot learn quickly in a new environment. Therefore, we introduce the federated learning model to realize the sharing of training experience between different datacenter environments.

3.3 Federated Reinforcement Learning Model in Multiple Datacenters

Service function chain orchestration decisions are learned from the interaction of the environment and the agent through reinforcement learning. In the cross-datacenter service orchestration scenario in this paper, a single datacenter has insufficient experience in orchestrating different types of services in different network environments, and needs to rely on the business processing experience of multiple datacenters. However, the business data of different datacenters may be under the jurisdiction of different companies, and it is difficult to collect training. Therefore, it is necessary to use the secure fusion of federated learning to achieve experience sharing and improve the reinforcement learning model proposed in the previous section. This section will introduce the generation, transfer, and correction process of federated reinforcement learning models.

After obtaining the reinforcement learning models of different datacenter environments, a federated reinforcement model is trained through the federated learning framework. The federated learning framework used in this work

is shown in Fig. 1. Each participant obtains its reinforcement learning model after local training. Model parameters are stored in a private Q-table, which is a map of state-action and reward in reinforcement learning. Q-table information in different training environments is encrypted and transmitted by homomorphic encryption. After the encrypted results are sent to the aggregation server, model parameters are decrypted and securely aggregated as a federated Q-table. In each iteration, the aggregation server sends the generated federated learning model back to each training environment for updating. With the fusion of local and federal model parameters, multiple datacenters can share decision-making experiences. In addition to sharing models in the training environment, our method can also serve as a basis for model training in the new environment and participating in subsequent training.

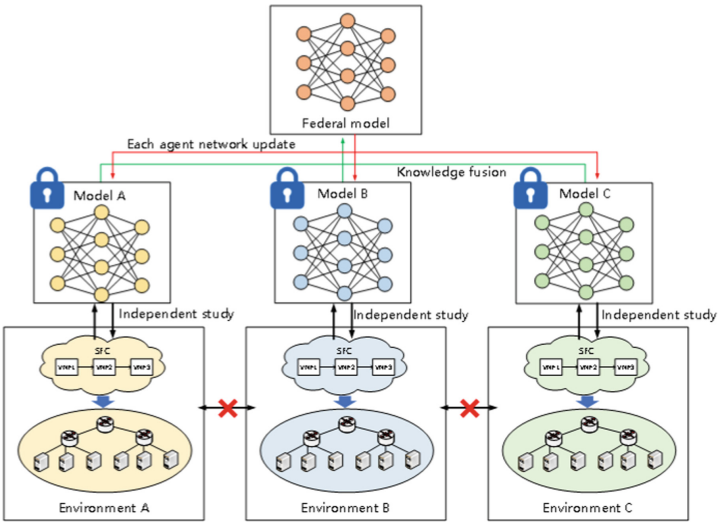

Fig. 1. The proposed federated learning framework.

In the generation process of the federated learning model, the placement action in the current state depends on the confidence of different datacenters. For example, in a certain state, the Q-table trained in the environment of datacenter A evaluates the Q-value of the current action as (5, 4, 4, 5, 6, 5, 5, 5, 5), and the Q-value in the environment of datacenter B is (3, 1, 10, 2, 2, 1, 1, 2, 10, 1). Then, it can be considered that datacenter B has higher confidence because of the more significant variance. However, the comparison by variance ignores the influence of extreme values. In contrast, information entropy is more suitable for measuring the uncertainty of information. In this work, we therefore use information entropy to define confidence.

For the main-body j, the confidence value under state x is defined as:

$$c_{xj} = -\frac{reward_{ij}}{\sum_{i=1}^{m} reward_{ij}} \cdot ln(\frac{reward_{ij}}{\sum_{i=1}^{m} reward_{ij}}) \tag{8}$$

where n is the number of the main-body, m is the action dimension (the number of nodes can be selected), $reward$ represents the return value. The confidence level w_{xj} is defined as shown in Eq. 9:

$$w_{xj} = \frac{(1 - c_{xj})}{\sum_{j=1}^{n} (1 - c_{xj})} \tag{9}$$

Assuming that there are k states in the reinforcement learning environment, and the Q-table is a $k \times m$ dimensional matrix, the federated model generation formula is shown in Eq. 10:

$$Q_{fl} = \sum_{j=1}^{n} Q_j \cdot \begin{pmatrix} w_{11}, w_{12}, \ldots, w_{1m} \\ \vdots \\ w_{k1}, w_{k2}, \ldots, w_{km} \end{pmatrix}^T \tag{10}$$

where Q_j is the main-body j participating in the training Q-table.

After the federated learning model is generated, the aggregation server will send the model to the training subjects in different environments to ensure the adaptability of the federated learning model.

The training subjects use Eq. 11 to update the local model so that the local model can learn more features.

$$Q_{new} = \frac{\alpha \cdot Q_{old} + Q_{fl}}{\alpha + 1} \tag{11}$$

where Q_{old} represents the original Q-table, Q_{new} represents the newly obtained Q-table, a represents the weight of the original Q-table. The federated model is sent to the original training environment and new environments to fuse models. Continue the training of reinforcement learning until the model converges. The iteration termination condition is defined as the difference between each new and the original Q-table is less than a pre-set threshold. The calculation by Euclidean distance is shown as:

$$\|Q_{old} - Q_{new}\| = \sqrt{(Q_{old} - Q_{new})(Q_{old} - Q_{new})^T} \leq \delta \tag{12}$$

In order to prevent leakage of the training data in different environments the federated learning framework needs to encrypt the Q table of the trained reinforcement learning. This process can ensure the safety and reliability of the transmission process as well as the privacy of the training data.

During the parameter transfer process of the federated learning model, we adopt homomorphic encryption to ensure the security of the model. Homomorphic encryption is a classic encryption algorithm. Homomorphic refers to a map

from an algebraic structure to a similar algebraic structure, which can keep all relevant structures unchanged. Since the result obtained by decrypting the homomorphic encrypted ciphertext after specific operations is consistent with the result obtained by decrypting the ciphertext and then performing specific operations, it can effectively ensure the confidentiality of data operations in the federated model fusion stage, so it is very suitable for federated learning scenarios. Compared with the method of secure multi-party computing, the data interaction using homomorphic encryption is less. Therefore, the communication overhead is less, and the efficiency of model fusion can be improved in federated learning scenarios that require multi-party participation.

The process of model transmission using homomorphic encryption is as follows: First, the aggregation server generates a homomorphic encryption public-private key pair, and distributes the public key to each participant of federated learning. Second, each participant transmits the calculation result to the aggregation server in the form of homomorphic ciphertext. The aggregation server performs summary calculations and decrypts the results based on private key and ciphertext. Finally, the aggregation server send the decrypted results to all the participants, and the participants update their model parameters according to the results. So far, the process of aggregation and distribution of a model based on homomorphic encryption has been completed. This process is repeated periodically to guarantee the performance of federated learning model. By using a secure federated fusion algorithm with homomorphic encryption technology in the model transmission and model fusion stages, participants of different datacenters are prevented from private information leakage at any stage of model training.

In order to ensure the adaptability of the federated learning model in the new environment, it is necessary to migrate and correct the obtained federated learning model to reduce the number of training times required to obtain the orchestration model in the new environment. Figure 2 shows the basic process of federated learning model migration and correction. First, the reinforcement learning models obtained in different training environments are fused to generate a federated model. Secondly, it is necessary to transfer the federated model to the original training environment and the new environment. The model in the training environment is fused with the federated model. The model fusion method is shown in Eq. (11) in the new environment, the federated model is used as the pre-trained model. Continue the training of reinforcement learning until the model converges. In practical applications, the above training and fusion steps are usually repeated several times.

4 Simulation Analysis

In this section, we describe the design of the experiments and present the analysis of empirical results.

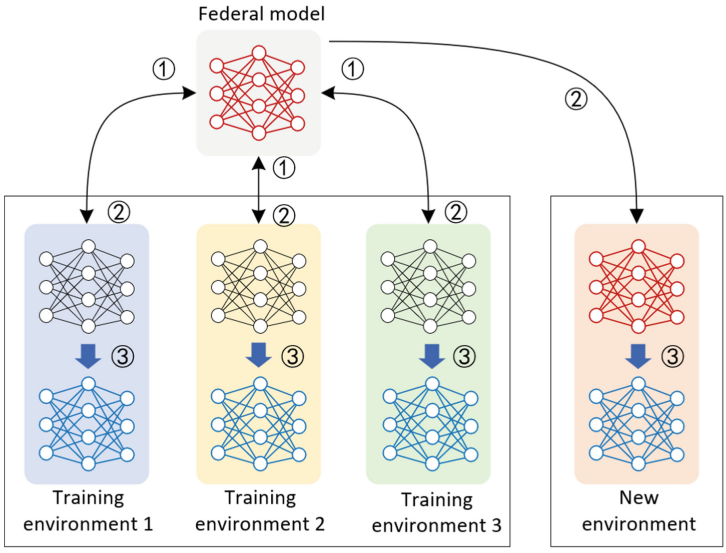

Fig. 2. Model migration and calibration process.

4.1 Design of Simulation Experiment

In order to verify the service orchestration effect of the federated reinforcement learning model proposed in this paper, it is necessary to design an appropriate simulation environment and orchestration tasks. We took a video analysis scenario as an example for simulation analysis. There are four VNFs in the video surveillance service that are placed and linked in the simulated network to provide users with video surveillance services, namely Motion Analyzer, Video Processor, Policy Decision and Mobile Proxy.

Three different training environments are designed in simulation, corresponding to different datacenters. Two are training environments with previous orchestration experience, and one is a new environment without orchestration experience. The simulation framework is shown in Fig. 3. The training processes of the three environments are independent of each other, and only after the federated reinforcement model is generated will the model information be exchanged.

The training processes of the three environments are independent of each other, and model information can only be exchanged after the federated reinforcement model has been generated. We sets 20 nodes to deploy the service function chain in each environment. The reliability and VNF load are different in each node. The cost of placing VNF is proportional to the spatial distance between nodes. The service function chain is to be arranged by four VNF serial compositions.

We train the reinforcement learning models in two environments independently to simulate the process of obtaining experience in the early stage. After models converge, training environments have orchestration experiences. The

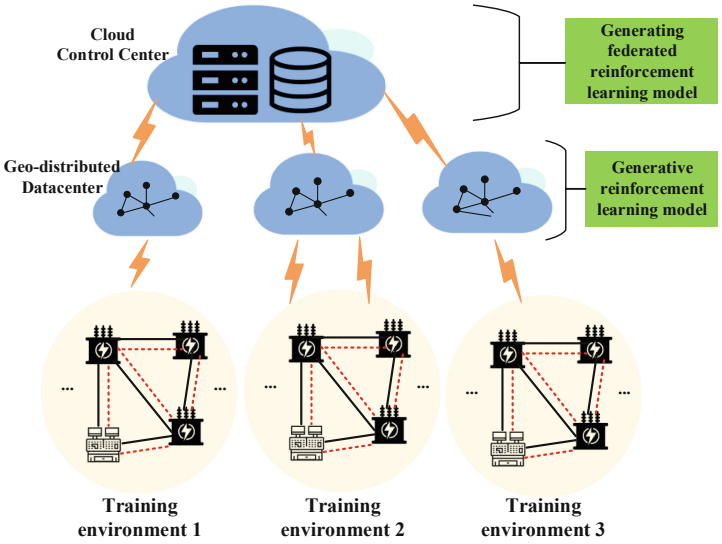

Fig. 3. Simulation framework.

training models are federated through the framework of federated learning. Then the generated federated model is sent to two training environments and a new environment for model fusion. We compare the proposed model with commonly used methods in terms of service reliability, resource overhead and efficiency, respectively. The VNF in the simulation is set up as a medium OpenStack VM with 2 vCPUs, 40 GB disk and 4 GB memory.

4.2 Empirical Results

We design three parts of simulation experiment for the service function chain orchestration method proposed in this paper. First, we analyze the convergence of the proposed federated reinforcement learning. Second, we verify the effectiveness of the proposed federated reinforcement learning framework in service function chain orchestration. Third, we verify the performance of the proposed service function chain orchestration method in jointly optimizing cost and reliability. The following is a detailed analysis of the experimental results.

Figure 4 shows the convergence of algorithm under different values of learning rate. Learning rate is one of the important hyperparameters in neural networks that affects the performance of federated reinforcement learning. As shown in Fig. 4, the model achieves the best performance when the learning rate is set as 0.01. When learning rate is set smaller as 0.001, The model converges at the local optimal solution and cannot reach a better reward. When the learning rate

is set too large as 0.1, the model will skip some of the learning process since the step size of the neural network exploration will be larger. At this time, the effect of the model is in an oscillating state, and the convergence performance is not good. Therefore, in subsequent experiments we set the learning rate to 0.01.

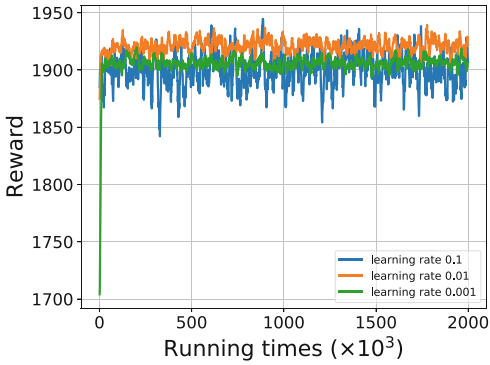

Fig. 4. Convergence performance of the model.

Figure 5 shows the training process of models in the new environment. The retrained Q-learning model converges after 800,000 iterations, and our method converges after 10,000 iterations. In this work, by comparing rewards of models during different iterations, we can prove that the convergence rate of reinforcement learning will be accelerated under the federated framework due to the experience of pre-training. In addition, the retrained Q-learning is trained based on random initialization parameters and local data. Therefore, at the beginning of the experiment, a low reward is obtained due to lack of experience in orchestration. And subsequent training is used to gradually obtain better orchestration decisions. While the federated reinforcement learning uses a model aggregated based on the orchestration experience of multiple datacenters. It has a certain orchestration experience at the beginning of the experiment, so better orchestration decisions can be obtained in early stage of training. As the training progresses, the optimal decision is gradually obtained.

To verify the effectiveness of the proposed federated learning framework in this work. We compare it with the independently retrained reinforcement learning model in the new and training environments. The simulation results of reliability and cost are shown in Fig. 6 and Fig. 7.

Figure 6 shows the comparison of the learning effects of our method (federated reinforcement learning model) and the retrained Q-learning model in the training environment. Regarding service reliability, our method can achieve high reliability and maintain stability at the beginning of training, while the retrained

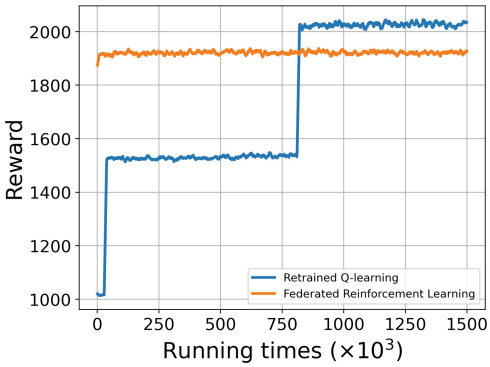

Fig. 5. Comparison of model convergence rates.

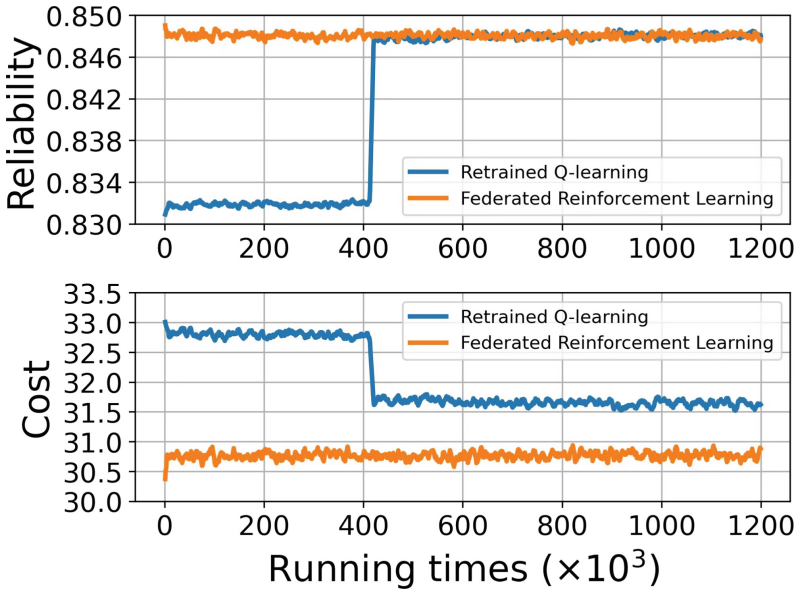

Fig. 6. Comparison of federated learning and reinforcement learning in training environment.

Q-learning model needs about 400,000 trainings to achieve high-reliability values. Regarding resource overhead, our method can consume lower costs, while the retrained Q-learning model still cannot reach the level that the federated model can achieve in a limited number of trainings. The reason is that our method integrates the training parameters of multiple environments, which can avoid a single environment training falling into local optimum. Generally speaking, the federated learning framework can achieve higher service reliability and consume fewer resources for training and decision-making in a training environment.

Figure 7 shows the comparison of the learning effects of our method and the retrained Q-learning model in the new environment. The emulation results show that our method can quickly reach a convergence state regarding training service reliability. Besides, the service reliability is maintained at a high value throughout the training process. In terms of resource overhead of the service function chain, our method can achieve rapid convergence and consume lower costs. The reason is that our method contains training experience obtained in other training environments, which achieve faster convergence when completing similar tasks. In general, our method for training and decision-making in the new environment can achieve higher service reliability and consume fewer costs.

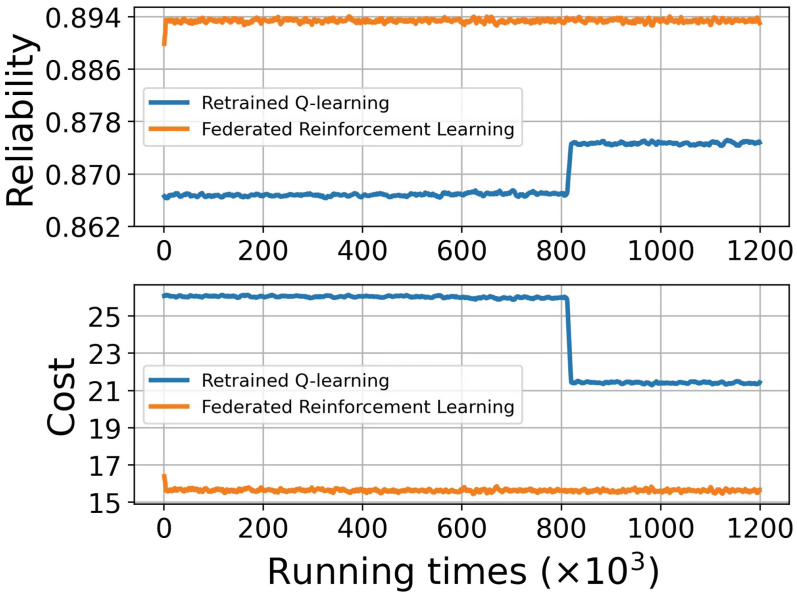

Fig. 7. Comparison of federated learning and reinforcement learning in the new environment.

To verify the performance of our proposed orchestration method, we select two greedy algorithms to compare the reliability and costs of service orchestration: the reliability-first greedy algorithm and the cost-first greedy algorithm. Table 1 shows experimental metrics in each case, which were statistically analyzed using average values over the 10 experiments. The best values of metrics in different cases are highlighted in bold.

Our proposed orchestration method based on federated reinforcement learning has significant reliability. In comparison with the reliability-first greedy algorithm, our method performs the best service reliability at a lower cost. Cost-first greedy algorithm leads to the lowest cost, and the overhead of our method is slightly higher. We can observe that the performance of our proposed method is better than other two methods, or it is close to the best value of metrics.

Table 1. Comparison of algorithm effects.

	Federated reinforcement learning model	Reliability-first greedy algorithm	Cost-first greedy algorithm
Service reliability	**0.993**	0.991	0.982
Cost	7.038	11.759	**7.000**

5 Conclusion

In this work, we provide an orchestration method for a reliable service function chain of datacenters to realize cloud-edge collaborative computing for large-scale data tasks. Specifically, we have developed an orchestration method based on federated reinforcement learning for the service function chain, which can improve the reliability of computing services while reducing cost. The proposed model is based on federated reinforcement learning. Training subjects in different geo-distributed datacenters can share training experiences through the aggregation model in cloud control center. Simulation results show that our proposed method can obtain a significant service orchestration decision for reliability improvement and cost reduction.

The computational difficulty will be greatly increased when cloud-edge collaborative computing architecture grows in complexity. In future work, we will explore using deep reinforcement learning models as basis leaners for increasingly complex business scenarios.

References

1. Tai, Y.C., Yen, L.H.: Network service embedding in multiple edge systems: profit maximization by federation. In: IEEE International Conference on Communications (ICC), pp. 1–6. IEEE (2021)
2. Jia, Y., Wu, C., Li, Z., et al.: Online scaling of NFV service chains across geo-distributed datacenters. IEEE/ACM Trans. Netw. **26**(2), 699–710 (2018). https://doi.org/10.1109/tnet.2018.2800400
3. Qing, H., Weifei, Z., Julong, L.: Virtual network protection strategy to ensure the reliability of SFC in NFV. In: Proceedings of the 6th International Conference on Information Engineering, pp. 1–5. ACM, New York (2017)
4. Wang, S., Zhou, A., Yang, M., et al.: Service composition in cyber-physical-social systems. IEEE Trans. Emerg. Topics Comput. **8**(1), 82–91 (2020). https://doi.org/10.1109/TETC.2017.2675479
5. Qu, L., Assi, C., Khabbaz, M.J., et al.: Reliability-aware service function chaining with function decomposition and multipath routing. IEEE Trans. Netw. Serv. Manag. **17**(2), 835–848 (2020)
6. Troia, S., Alvizu, R., Maier, G.: Reinforcement learning for service function chain reconfiguration in NFV-SDN metro-core optical networks. IEEE Access **7**, 167944–167957 (2019). https://doi.org/10.1109/ACCESS.2019.2953498

7. Shah, H.A., Zhao, L.: Multiagent deep-reinforcement-learning-based virtual resource allocation through network function virtualization in internet of things. IEEE Internet Things J. **8**(5), 3066–3074 (2020). https://doi.org/10.1109/JIOT.2020.3023111
8. Liu, X., Yu, J., Feng, Z., et al.: Multi-agent reinforcement learning for resource allocation in IoT networks with edge computing. China Commun. **17**(9), 220–236 (2020). https://doi.org/10.23919/JCC.2020.09.017
9. Chen, H.M., Chen, S.Y., Wang, S.K., et al.: Designing a reinforcement learning approach for the NFV orchestration system with energy saving optimization. In 2022 8th International Conference on Applied System Innovation (ICASI), pp. 98–10. IEEE (2022)
10. Huang, H., Zeng, C., Zhao, Y., et al.: Scalable orchestration of service function chains in NFV-enabled networks: a federated reinforcement learning approach. IEEE J. Sel. Areas Commun. **39**(8), 2558–2571 (2021). https://doi.org/10.1109/JSAC.2021.3087227
11. Zhang, P., Wang, C., Jiang, C., et al.: Deep reinforcement learning assisted federated learning algorithm for data management of IIoT. IEEE Trans. Industr. Inform. **17**(12), 8475–8484 (2021). https://doi.org/10.1109/TII.2021.3064351
12. Nguyen, D.C., Ding, M., Pathirana, P.N., et al.: Federated learning for internet of things: a comprehensive survey. IEEE Commun. Surv. Tutor. **23**(3), 1622–1658 (2021). https://doi.org/10.1109/JIOT.2022.3170449
13. Dieye, M., Ahvar, S., Sahoo, J., et al.: CPVNF: cost-efficient proactive VNF placement and chaining for value-added services in content delivery networks. IEEE Trans. Netw. Service Manag. **15**(2), 774–786 (2018). https://doi.org/10.1109/TNSM.2018.2815986
14. Sang, I.K., Kim, H.S.: A VNF placement method based on VNF characteristics. In: International Conference on Information Networking, Jeju Island, pp. 864–869. IEEE (2021)
15. Yang, Z., Chen, B., Dai, M., et al.: VNF placement for service chaining in IP over WDM networks. In: Asia Communications and Photonics Conference, Hangzhou, pp. 1–3. IEEE (2018)
16. Kar, B., Wu, E.H.K., Lin, Y.D., et al.: Energy cost optimization in dynamic placement of virtualized network function chains. IEEE Trans. Netw. Service Manag. **15**(1), 372–386 (2018). https://doi.org/10.1109/TNSM.2017.2782370
17. Varasteh, A., Madiwalar, B., Bemten, A.V., et al.: Holu: power-aware and delay-constrained VNF placement and chaining. IEEE Trans. Netw. Service Manag. **18**(2), 1524–1539 (2021). https://doi.org/10.1109/TNSM.2021.3055693
18. Quang, P., Hadjadj-Aoul, Y., Outtagarts, A.: On using deep reinforcement learning for VNF forwarding graphs placement. In: 11th International Conference on Network of the Future, Bordeaux, pp. 126–128. IEEE (2020)
19. Pei, J., Hong, P., Pan, M., et al.: Optimal VNF placement via deep reinforcement learning in SDN/NFV-enabled networks. IEEE J. Sel. Areas Commun. **38**(2), 263–278 (2020). https://doi.org/10.1109/JSAC.2019.2959181
20. Rui, L., Chen, X., Gao, Z., et al.: Petri net-based reliability assessment and migration optimization strategy of SFC. IEEE Trans. Netw. Service Manag. **18**(1), 167–181 (2020). https://doi.org/10.1109/tnsm.2020.3045705