electronics

MDPI

Article

Reinforcement-Learning-Assisted Service Function Chain Embedding Algorithm in Edge Computing Networks

Wei Wang [1], Shengpeng Chen [2], Peiying Zhang [2,3] and Kai Liu [4,5,*]

[1] School of Information Engineering, Guangzhou Panyu Polytechnic, Guangzhou 511483, China; wangw@gzpyp.edu.cn
[2] Qingdao Institute of Software, College of Computer Science and Technology, China University of Petroleum (East China), Qingdao 266580, China; z24070039@s.upc.edu.cn (S.C.); zhangpeiying@upc.edu.cn (P.Z.)
[3] Key Laboratory of Computing Power Network and Information Security, Ministry of Education, Shandong Computer Science Center (National Supercomputer Center in Jinan), Qilu University of Technology (Shandong Academy of Sciences), Jinan 250013, China
[4] State Key Laboratory of Space Network and Communications, Tsinghua University, Beijing 100084, China
[5] Beijing National Research Center for Information Science and Technology, Tsinghua University, Beijing 100084, China
* Correspondence: liukaiv@tsinghua.edu.cn

Abstract: Edge computing networks are critical infrastructures for processing massive data and providing instantaneous services. However, how to efficiently allocate resources in edge computing networks to meet the embedding requirements of service function chains has become an urgent problem. In this paper, we model the resource allocation problem in edge computing networks as a service function chain embedding problem model, aiming to optimize the resource allocation through reinforcement learning algorithms to achieve the goals of low cost, high revenue, and high embedding rate. In this paper, the basic concepts of edge computing network and service function chain are elaborated, and the resource allocation problem is transformed into a service function chain embedding problem by establishing a mathematical model, which provides a foundation for the subsequent algorithm design. In this paper, a service function chain embedding algorithm based on reinforcement learning is designed to gradually optimize the resource allocation decision by simulating the learning process. In order to verify the effectiveness of the algorithm, a series of simulation experiments are conducted in this paper and compared with other algorithms. The experimental results show that the service function chain embedding algorithm based on reinforcement learning proposed in this paper exhibits superior performance in resource allocation. Compared with traditional resource allocation methods, the algorithm achieves significant improvement in terms of low cost, high revenue, and high embedding rate.

Keywords: edge computing; resource allocation; service function chaining; distributed reinforcement learning

Citation: Wang, W.; Chen, S.; Zhang, P.; Liu K. Reinforcement-Learning-Assisted Service Function Chain Embedding Algorithm in Edge Computing Networks. *Electronics* **2024**, *13*, 3007. https://doi.org/10.3390/electronics13153007

Academic Editors: Javid Taheri and Christos J. Bouras

Received: 28 June 2024
Revised: 24 July 2024
Accepted: 28 July 2024
Published: 30 July 2024

1. Introduction

With the rapid development of the Internet of Things (IoT) and cloud computing technology, edge computing, as a new computing paradigm, has increasingly attracted the attention of researchers. In the construction of smart cities, edge computing has played an important role in data processing, service provision, etc. However, achieving efficient resource allocation and management on edge devices with limited resources is an important challenge in smart city networks. This article aims to solve the problem of resource allocation and management in smart cities through reinforcement learning (RL) algorithms.

Edge computing refers to the computing mode of data processing and services near data sources or users. With the expansion of urban scale and the growth of data volume,

traditional centralized data processing methods can no longer meet the needs of real-time and low latency. Edge computing, as a solution, demonstrates tremendous development potential. It facilitates the migration of computing and storage resources towards IoT devices, effectively bridging the latency gap between the IoT system and the underlying cloud computing framework. Edge computing has a broad application prospect in the construction of smart cities [1]. It can help realize the intelligence, efficiency, and convenience of cities and improve the quality of life and happiness of urban residents. For example, edge computing can help to realize intelligent transportation systems. Through real-time analysis of data collected by road sensors, traffic flow can be optimized and congestion can be reduced. In terms of urban security, edge computing monitors the security situation of cities in real time through video monitoring and image recognition technology, and finds and handles security incidents in a timely manner. The collaboration of heterogeneous edge computing paradigms significantly enhances the functionality and efficiency of smart city applications [2]. By integrating cloudlet, mobile edge computing, and fog computing, it is possible to streamline resource utilization and improve the performance of urban systems. For instance, through blockchain-based smart contracts and software-defined networks, real-time traffic management can be optimized, reducing congestion and improving urban mobility. Additionally, edge computing can bolster city security through continuous surveillance and rapid incident response, enhancing the overall safety and quality of life for residents. At the same time, the use of intelligent access control systems and intelligent alarm systems by edge computing can improve the level of urban security. In addition, smart healthcare, smart education, smart tourism, etc. have greatly improved the convenience and efficiency of public services.

The development of the internet has brought convenience to people's lives [3]. However, with the introduction of new protocols and technologies, the Internet has become rigid and unable to meet users' needs for service diversity [4,5]. To address this issue, T. Anderson et al. [6] proposed network virtualization technology, which can allocate network resources rationally based on dynamically changing user needs. Network virtualization technology is considered one of the most promising technologies for the future internet [7]. Currently, due to the need to comprehensively consider multiple indicators of mapping results in practical applications [8], single-domain virtual network mapping has gradually become unable to meet people's business needs [9]. Therefore, the concept of multidomain network mapping emerged. It also plays an important role in the construction of smart cities. It can integrate network resources from different fields to form a unified virtual network, enabling the interconnection of data and services, thus facilitating collaboration between various fields. Secondly, by connecting various sensors, devices, and services to the virtual network, real-time data collection, analysis, and processing can be realized. This helps to detect and solve problems in urban operations in a timely manner, improving the response speed and service quality of the city. In addition, multidomain virtual network mapping also helps to protect the data security of the city. By storing data in the virtual network, it can strengthen the supervision and protection of data and prevent data leakage and attacks. At the same time, multidomain virtual network mapping can also achieve data sharing and exchange, promoting data circulation and cooperation between various fields.

RL is an important branch of machine learning that aims to explore new actions using known actions [10,11]. The basic idea is to enable agents to learn how to make optimal decisions through interaction with the environment. In RL, agents continuously try different behaviors and adjust their behavior strategies based on the reward information feedback from the environment, ultimately achieving the optimal decision goal. In recent years, RL combined with deep neural networks has provided powerful tools for solving complex decision-making problems.

The main contribution of this paper is to propose a new resource allocation and management method by combining the edge computing network, RL algorithm, and service function chain embedding algorithm. The specific innovation points are as follows:

- Distributed-based reinforcement learning algorithm: We designed a RL algorithm that can make dynamic resource allocation decisions based on the real-time state of the system. Through a distributed architecture, the algorithm can handle large-scale smart city networks and has good scalability.
- Embedded algorithm for service function chain: For the embedding problem of the service function chain, we propose an optimized algorithm. This algorithm can make intelligent mapping decisions based on the requirements of service functions and the characteristics of physical resources, thereby improving the overall performance and stability of the system.

Through conducting three sets of comparative experiments, the significant advantages of the algorithm proposed in this article in improving long-term average income, long-term service function chain request (SFCR) acceptance rate, and long-term revenue–cost (R/C) ratio were verified.

The structure of this paper is as follows: The first part briefly introduces the relevant background of edge computing and smart cities. The second part is an introduction to the relevant work of this paper. The third part introduces the established edge computing network model and service function chain request model. The fourth part provides a detailed introduction to the algorithm implementation of distributed reinforcement learning. The fifth part presents the performance of the algorithm in simulated environments and discusses and compares it. The final section summarizes the main work and future research directions of this paper.

2. Related Work

2.1. Research Status of Edge Computing Networks

With the rapid development of technologies such as the Internet of Things and cloud computing, edge computing networks were widely used in industry, medical care, transportation, smart home, and other fields. Many compute-intensive and time-delay-sensitive applications (such as unmanned driving, image recognition, and natural language processing) experienced an explosive growth of service traffic, which placed high requirements on the computing power of terminal devices. However, due to the limitation of network resources and computing resources, resource allocation problems became increasingly prominent. In order to solve the problem of bandwidth resource optimization, many scholars at home and abroad proposed various methods to reduce the delay of data transmission and optimize the bandwidth resources of mobile networks. Mobile edge computing (MEC) technology came into being. MEC aimed to meet users' demand for high resource utilization and high reliability.

Abdullaev et al. [12] designed a new Task Offloading and Resource Allocation in IoT-based MEC using the Deep Learning with Seagull Optimization (TORA-DLSGO) algorithm. The resource management issues in the MEC server were solved, enabling optimal offloading decisions to minimize system costs. Chen [13] considered the joint optimization of computation offloading and task caching in a cellular network. It allowed users to proactively cache or offload their tasks at the MEC server. Liu [14] considered the impact of multiserver scenarios and task priorities in large networks and proposed a distributed unsupervised learning-based offloading framework for task offloading and server allocation. Gao [15] proposed a task scheduler and exploited a Task Deployment Algorithm (TDA) to obtain an optimal virtual machine deployment scheme; however, computing migration was not considered, resulting in an inability to respond to problems when demand exceeded. Yang et al. [16] proposed an energy-sensitive binary offloading algorithm for reconfigurable-intelligent-surface-assisted wireless powered mobile edge computing, aiming to optimize the balance between computational efficiency and energy consumption. Liu [17] developed a multiagent reinforcement learning framework, and an independent learner-based multiagent Q-learning (IL-based MA-Q) algorithm was proposed. Wen [18] proposed a task assignment algorithm combining a genetic algorithm

and an ant colony optimization algorithm. The algorithm prolonged the lifetime of the network and improved the efficiency and advantages of energy saving and load balancing.

2.2. Research Status of Network Resource Allocation Algorithms

Network resource allocation algorithms have always been the focus of research, and in recent years, with the development of deep reinforcement learning, network resource allocation algorithms based on deep reinforcement learning have also been widely recognized. Li [19] proposed a resource management framework based on distributed reinforcement learning (RL), which significantly reduced power consumption and content transmission delay. In [20], to address the joint optimization problem of D2D communication mode selection and resource allocation in MMwave and cellular HCNS, a distributed multiagent deep Q-network algorithm was proposed, and the reward function was redefined according to the objective to reduce signaling overhead. In [21], a finite-state Markov model based on fading characteristics achieved reasonable resource allocation by capturing user throughput in cellular network interactions and allowing all mobile users to efficiently share the same spectrum resource simultaneously, thus improving fairness for users with lower transmission costs in the mobile edge computing model. Li [22] introduced an intelligent offloading mechanism for mobile edge computing based on content caching. Initially, a compute offload network framework based on content cache in mobile edge computing was designed. Then, by deducing sufficient and necessary conditions, the optimal contract was designed to obtain the computing strategy under joint task unloading, resource allocation, and intelligent mechanisms. Xu [23] proposed a resource allocation algorithm based on PDQN, a deep reinforcement learning algorithm, to address the resource allocation problem in collaborative cloud–edge computing with dynamic user demand and multiple cloud service pricing models. Sun [24] designed incentives and cross-server resource allocation in blockchain-driven MEC, achieving a decentralized, immutable, secure, and fair resource allocation mechanism. Wang [25] regarded offload decisions, resource allocation, and content caching strategies as an optimization problem considering total network revenue and proposed an alternate direction algorithm based on a multiplier to solve the optimization problem. Additionally, Zhang et al. [26] investigated cooperative resource allocation problems for multilevel services in MEC networks and proposed a cooperative resource allocation (FD-CRA) algorithm based on federated learning and deep Q network (DQN). He [27] proposed an MHRL method for dynamically adaptive management of vehicle resources, making correct and effective resource allocation decisions for vehicle requests and addressing some challenges encountered by vehicles in dynamic environments. Zhang [28] proposed a computation resource allocation scheme for VANETs based on deep reinforcement learning networks in the context of MEC scenarios. This scheme effectively allocated computing resources for VANETs in edge computing environments, demonstrating excellent network performance with low overhead and latency.

2.3. Summary of Related Work

While some excellent work had been performed in resource allocation within edge computing networks, there had been relatively little consideration of network resource allocation in existing research. What set this study apart was its focus on optimizing resource allocation within edge computing networks, employing a virtual network mapping algorithm based on distributed reinforcement learning. Specifically, we integrated edge computing networks with RL algorithms and service function chaining embedding algorithms, enabling the algorithm to dynamically adjust resource allocation in real time according to network dynamics, thus achieving equitable traffic distribution across nodes and links.

3. Network Modeling

The edge computing network model, as a crucial component of multidomain physical network mapping, provides an abstract representation of edge computing networks. This model integrates service function chain embedding technology, effectively transforming

edge computing resources into resources that meet the specific requirements of service function chains. By optimizing key resource attributes, this model significantly enhances the data processing efficiency of IoT devices in smart cities and meets strict requirements for security and privacy. This comprehensive optimized model is applied in multidomain network architectures, not only providing advantages in real-time data processing and efficiency but also strengthening data security and privacy protection, thereby providing reliable foundational support for smart cities and IoT applications. Our goal is to strategically allocate resources within the edge computing network, aiming to increase the acceptance rate of terminal requests while reducing costs and improving average returns. To achieve this, we have established relevant system models, resource allocation constraints, and evaluation metrics.

3.1. Modeling Edge Computing Networks

In the edge computing network model, the integration of multidomain physical networks with service function chain (SFC) embedding algorithms offers a highly specialized and precise resource management strategy. Edge computing networks reduce data transmission latency significantly by deploying computational resources closer to end-users and IoT devices, thereby enhancing service response and data processing efficiency. Moreover, the design of multidomain physical networks allows dynamic distribution of network resources across various geographic and logical domains, facilitating flexible scheduling of edge computing resources. The service function chain embedding algorithm, building on this, virtualizes physical resources distributed across multiple domains, effectively supporting diverse network services. This holistic application not only improves resource utilization but also enhances the scalability and security of the network, especially in managing complex data and service demands in smart cities. SFC request refers to a type of request issued within a computer network with the aim of executing a specific service function chain. When users or applications require completing specific tasks or obtaining particular services, they send such requests. SFC requests traverse along the network path and pass through a series of network nodes en route. Each node performs specific service functions, which are organized in a certain sequence to form the service function chain, ultimately meeting the requester's needs. Consistent with prior work [1], we abstract the underlying physical network as an undirected graph, denoted as $G^S = \{N^S, L^S, AN^S, AL^S\}$, where N^S represents the set of underlying nodes, L^S represents the set of underlying links, A$N^S$and AL^S respectively represent the attributes associated with the underlying nodes and links. Similarly, we use another undirected graph to represent the service function chain embedding, denoted as $G^C = \{N^C, L^C, CN^C, CL^C\}$, where N^C represents the set of nodes within the service function chain, L^C represents the set of links within the service function chain, $CN^C$and CL^C, respectively, represent the attributes associated with the nodes and links within the service function chain. Each time a request is issued, the underlying network resources need to be greater than the network resources required by the request, i.e., $G^C\{N^C, L^C\} \longrightarrow G^S\{N^S, L^S\}$ where and $L^C \in L^S$. Figure 1 illustrates an edge computing network model based on virtual architecture. The virtual network request comprises multiple layers, with each distinct layer describing various network functionalities and services. Moreover, these layers are utilized to specify the necessary resources and services required to fulfill specific demands.

3.2. Resource Properties and Constraint Conditions

- Resource Properties

In the SFC request model, optimizing physical network performance and meeting user requirements is achieved through the reasonable setting and allocation of resource attributes. Based on the computational requirements of tasks, CPU core numbers are determined to define the required computational capacity; network transmission rates are configured according to the required data transmission capacity, thereby determining network bandwidth requirements or constraints; network latency, transmission delay,

processing delay, etc., are set based on the timeliness requirements of tasks to ensure the real-time processing of tasks; and different security requirements can be addressed by introducing security mechanisms such as data encryption, identity authentication, etc., thereby protecting network security and data privacy. During the connection of virtual networks and physical networks, considerations such as bandwidth constraints, latency constraints, resource capacity constraints, reliability constraints, etc., are necessary. These constraints can be set based on the resource attributes mentioned above to ensure that physical network requirements are met while optimizing overall performance.

- Constraint Conditions

The embedding of the SFC involves mapping the service function requests (SFR), which include service function nodes (SFN) and service function links (SFL), onto physical network resources. These physical resources encompass data centers, network nodes, and other types of infrastructure. Each SFN may need to be mapped onto a physical node with sufficient resources. For this purpose, we define the candidate physical node attribute $candi_p^n$, represented as:

$$candi_p^n = \{p^{n_1}, p^{n_2}, ..., p^{n_m}\} \quad (1)$$

where p^{n_i} represents an available physical node. This attribute ensures that the SFN can be mapped to an appropriate physical node.

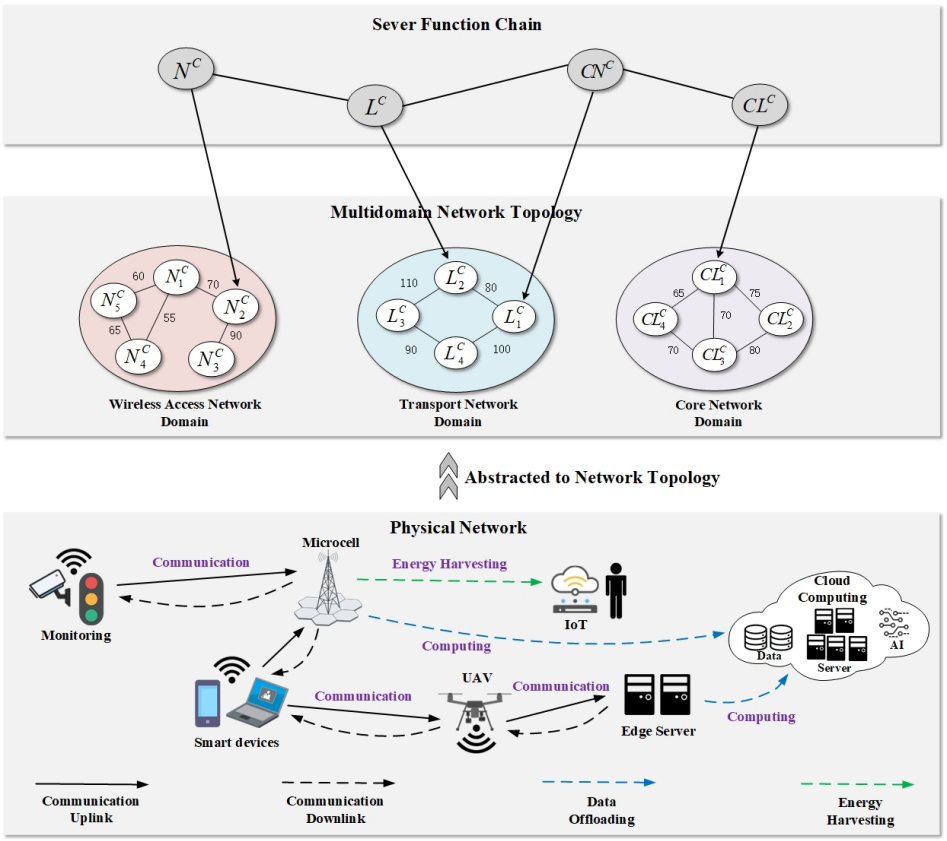

Figure 1. Mapping framework of the multidomain physical network.

In addition, we must ensure that the location constraints for service function nodes are satisfied, meaning each SFN can only be mapped to one physical node. Since different SFNs may perform different subservices, stringing them together can meet the complete service requirements of the end user. For the same end-user request, each SFN can only be mapped to one physical node. To connect these service function nodes, the SFL can be mapped to one or more physical links. These constraints are defined as follows:

$$\sum_{k=1}^{|SFN|} \lambda_{pn}^{SFN_k} = 1, SFN_k \in GR_i \tag{2}$$

$$\sum_{k=1}^{|SFL|} \mu_{pl}^{SFL_k} \geq 1, SFL_k \in GR_i \tag{3}$$

where $\lambda_{pn}^{SFN_k}$ and $\mu_{pl}^{SFL_k}$ are binary variables. If $\lambda_{pn}^{SFN_k} = 1$, it indicates that the service function node SFN_k is mapped to the physical node pn. If the service function node SFN_k is not mapped to the physical node pn, then $\lambda_{pn}^{SFN_k} = 0$. Similarly, when $\mu_{pl}^{SFL_k} = 1$, it indicates that the service function link SFL_k is mapped to the physical link pl; when $\mu_{pl}^{SFL_k} = 0$, it indicates that the service function link SFL_k is not mapped to the physical link pl. $|SFN|$ represents the number of service function nodes in the end-user function request GR_i, and $|SFN|$ represents the number of service function links.

Beyond location constraints, the end user function requests must also follow resource constraint rules. With the candidate physical nodes determined, service function nodes can only be mapped to physical nodes that meet their resource requirements. Specifically, the CPU resource capacity of the target physical node should be no less than the CPU resource demand of the service function node. The storage resource capacity of the target physical node should be no less than the storage resource demand of the service function node. These resource constraints are defined as follows:

$$CPU(pn) \geq CPU(SFN_k), if \lambda_{pn}^{SFN_k} = 1, k = 1, 2, ..., |SFN| \tag{4}$$

$$STO(pn) \geq STO(SFN_k), if \lambda_{pn}^{SFN_k} = 1, k = 1, 2, ..., |SFN| \tag{5}$$

where $CPU(pn)$ and $STO(pn)$ represent the CPU resource capacity of the target physical node and the storage resource capacity of the target physical node, and $CPU(SFN_k)$ and $STO(SFN_k)$ represent the CPU resource demand of the service function node and the storage resource demand of the service function node, respectively.

Service function request links can only be mapped to physical links that meet their resource demands, which are defined as

$$BW(pl) \geq BW(SFL_k), if \lambda_{pn}^{SFL_k} = 1, k = 1, 2, ..., |SFL| \tag{6}$$

where $BW(pl)$ and $BW(SFL_k)$ represent the bandwidth resource capacity of the target physical node and the storage resource capacity of the target physical node.

3.3. Algorithmic Evaluation Metrics

In order to assess the resource utilization efficiency and performance of the algorithm, we have established a set of evaluation metrics aimed at comparing with previous research results, thereby demonstrating the effectiveness of the proposed algorithm in our paper.

- The long-term acceptance ratio

Let $Acc_num(G^C, t)$denote the number of *SFC* requests received at time t, $Arr\_num(G^C, t)$ denote the total number of requests arriving at the *SFC* up to time t. The calculation formula for the long-term acceptance rate is as follows:

$$ACR = limT \to \infty \frac{\sum_{t=0}^{T} Acc\_num(G^C, t)}{\sum_{t=0}^{T} Arr\_num(G^C, t)} \tag{7}$$

- The long-term average revenue

The ultimate goal of the service function chain embedding algorithm is to improve network performance and revenue. Similar to previous literature [3], we define the long-

term average revenue as the limit of the average revenue as time T tends to infinity, which can be expressed as

$$R = limT \to \infty \frac{\sum_{t=0}^{T} R(G^C, t)}{T} \tag{8}$$

- The long-term average revenue–cost ratio (R/C)

According to the cost calculation formula in [3], the calculation formula for R/C can be expressed as the following formula:

$$R/C = limT \to \infty \frac{\sum_{t=0}^{T} R(G^C, t)}{\sum_{t=0}^{T} Cost(G^C, t)} \tag{9}$$

3.4. System Models

In this subsection, we define the reinforcement learning model as an MDP, consisting of the state space $\mathcal{S}$, the action space $\mathcal{A}$, and the reward $\mathcal{R}$.

- ***State Space:*** The system state represents the real-time status of the physical network, including real-time computing resources, bandwidth, and other resource information, as well as real-time node latency, network topology, and other network information. Network state s_t can be defined as $s_t = \{CPU_t, BW_t, DLY_t, TLY_t\}$, where CPU_t represents the real-time available computing resources, BW_t denotes the network bandwidth, DLY_t denotes the node latency, and TLY_t describes the network topology.
- ***Action Space:*** The action of the agent is the SFC mapping decision taken at a certain moment t, including the SFN embedding scheme and the SFL embedding scheme, defined as $\mathcal{A} = \{\mathcal{A}_N, \mathcal{A}_L\}$, where $\mathcal{A}_N$ is a sequence of embedding scheme for SFN, and $\mathcal{A}_L$ is for SFL. The embedding scheme is expressed using the following formula:

$$\{\mathcal{A}_N, \mathcal{A}_L\} = \{(\lambda_{pN_i^s}^{t,SFN_i}), (\mu_{pL_j^s}^{t,SFL_j})\} \longleftarrow \begin{cases} \forall i \in [1, |SFN|] and \sum_i^{|SFN|} \lambda_{pN_i^s}^{t,SFN_i} = |SFN| \\ \forall j \in [1, |SFL|] \end{cases} \tag{10}$$

  where $\lambda_{pN_i^s}^{t,SFN_i}$ represents the decision of SFN mapping, if $\lambda_{pN_i^s}^{t,SFN_i} = 1$, and denotes the ith VNF in the SFC mapping in the physical node pN_i^s, and $\lambda_{pL_j^s}^{t,SFL_j}$ represents the decision of SFL mapping, if $\lambda_{pL_j^s}^{t,SFL_j} = 1$, and denotes the jth SFL mapping in the physical link pL_j^s, and if $\lambda_{pN_i^s}^{t,SFN_i} = 0$ or $\lambda_{pL_j^s}^{t,SFL_j} = 0$, this is an embedding failure that might be caused by current physical node resources being less than the VNF resource requirements or insufficient link bandwidth. After all VNFs in SFC have completed mapping, then one can start to consider mapping virtual network links to physical networks.
- ***Reward:*** The reward of an intelligent agent r_t is the feedback signal value obtained from the action taken, which is used to guide the optimization direction of the agent. In this work, we consider three factors, including long-term acceptance ratio (ACR), long-term average revenue (R), and long-term average revenue–cost ratio (R/C). The reward is computed using the following formula:

$$r_t = \alpha \cdot ACR + \beta \cdot R + \eta \cdot R/C \tag{11}$$

  where α, β, and η are weights used to adjust the significance of different signal values.

4. Algorithm

In this section, we propose a distributed reinforcement learning (DRL)-based algorithm, which includes an agent and an environment. We use DRL to assist the SFC embedding to the edge computing network. Using this algorithm, we compute the embedding scheme of the SFC after all virtual nodes are embedded. We first extract the environmental state matrix of the physical network as input for the agent. Secondly, combined with the

DRL training paradigm, the agent calculates candidate nodes and links based on a series of relevant constraints proposed above.

4.1. Parameter Settings

Extracting the attributes resources of the nodes to construct the feature matrix is a critical step. The feature matrix is an important influencing factor for computing the SFC embedding scheme. We extracted the following four essential and significant attributes of nodes:

1. $CPU(N_i^s)$: The computing resources of the physical node N_i determine its carrying capacity, and the node with higher computing resources is better capable of meeting user requirements.
2. $SUM_{bw}(N_i^s)$: Each physical node has at least one link connected to it. $SUM_{bw}(N_i^s)$ denotes the sum of the bandwidth of all the links connected to node N_i^s. When the $SUM_{bw}(N_i^s)$ of a node is higher, the virtual nodes it hosts can have better link embedding options.
3. $D(N_i^S)$: The delay performance of a physical node reflects how fast it can process Virtual Network Functions (VNFs). Nodes with lower delay can handle VNFs with stringent delay requirements.
4. $Deg(N_i^S)$: Degree denotes the total number of links connected to node N_i^s. Physical nodes with more adjacent links can have a higher possibility of successful link embedding.

4.2. Environment Perception

Combining the four features mentioned in Section 4.1, they can be represented synthetically using a feature vector, whose structure is depicted in the equation below:

$$V_i = \left\{ CPUbig(N_i^s big), SUM_{bw} big(N_i^s big), D(N_i^S big), Deg(N_i^S big) \right\} \tag{12}$$

Following this, multiple feature vectors can be combined to form a feature matrix, serving as the training environment for the local agent, with specific features as follows:

$$M = \left[V_1^T, V_2^T, ..., V_i^T, ..., V_n^T \right] \tag{13}$$

Then, using the extracted state matrix from Equation (13) as input for the agent, calculate the available resources of each node in the physical network through convolution operation. We designed a five-layer policy network, including an input layer, convolution layer, softmax layer, filtering layer, and output layer, to serve as the local agent. The convolution operation is defined by the following formula:

$$h_i^c = \omega v_i + b \tag{14}$$

where h_i represents the ith output of the convolutional layer, signifying the vector of available resources of the node. Moreover, ω denotes the weight, and b denotes the bias.

Subsequently, h_i is utilized as input to the softmax layer to produce the probability p of each node being embedded by the requested node. The calculation formula for p is outlined as follows:

$$p_i = \frac{e^{h_i^c}}{\sum_k^N e^{h_k^c}} \tag{15}$$

where N denotes the total number of physical nodes, while p_i signifies the probability that the ith node is embedded by the VNF. Once the probability p of a node being embedded is determined, the cross-entropy loss can be calculated. The loss is computed using the following formula:

$$L(p) = -\sum_i^N log(p_i) \tag{16}$$

4.3. Local Training

Algorithm 1 delineates the aforementioned process. Line 2 signifies the initialization of the model by the local agent based on the received global parameters. Lines 3–5 represent the VNF embedding phase, while lines 7–9 pertain to the link embedding phase. Line 11 calculates the reward according to the embedding scheme. Lines 17–20 encompass the SFC scheduling phase. Ultimately, the trained parameters θ are uploaded in line 24. The worst complexity for a single SFC request is $\mathcal{O}(|\mathcal{S}| \cdot |\mathcal{A}| + |N^c| \cdot |N^s| + |L^c|)$, where $|N^s|, |N^c|, |L^c|$ represents the set's size of the underlying nodes, the nodes, and links within the SFC, respectively.

Algorithm 1 Local Training

Input: $G^S, G^C, epoch$;
Output: Probability of SFC being embedded;

1: **while** iteration $<$ epoch **do**
2: **for** SFC $\in$ TrainingSet **do**
3: M = getFeatureMatrix(n_i); //Extract the feature matrix of the substrate network.
4: p = getProbability(M); //Get embedded probabilities.
5: host = select(p); //Select the physical node carrying SFN according to probability.
6: getGradient(host);
7: **if** is Mapped($\forall SFN_i \in SFC$) **then**
8: Virtual links embedding;
9: **end if**
10: **if** isMapped($\forall SFN_i \in G^C, \forall SFL_i \in G^C$) **then**
11: Calculate reward;
12: **else**
13: Clear Gradient;
14: **end if**
15: **end for**
16: **for** $n_i \in N^S$ **do**
17: **if** needSchedule(n_i) **then**
18: host = ReEmbedding(n_i, p);
19: Virtual links embedding;
20: **end if**
21: **end for**
22: iteration++;
23: **end while**
24: Upload parameter θ

5. Experimental Analysis

In this section, we conducted simulation experiments to showcase the performance of the algorithm introduced in this paper.

5.1. Experimental Environment and Parameter Settings

The simulation experiment platform utilizes VS Code 2019 + Python 3 + TensorFlow 1.0, with hardware specifications including an Intel(R) Core (TM) i7-8565U CPU and 8GB of RAM.

We programmatically simulated a medium-sized edge computing network consisting of 100 nodes and about 600 links, and the detailed physical network parameters and their values are shown in Table 1.

We utilize reinforcement learning to assist SFC embedding in the edge computing network, which can be trained and tested according to different target types. In each training cycle, it tries to receive SFCRs and allocate physical resources to them, as well as adjust the model weights to maximize the selected targets according to the target type. Moreover, in the simulation experiments, physical nodes are selected as mapping targets based on the provided edge computing networks, VNFs, in order to match the real scenario.

This selection can be random or probabilistic. Our simulation experiments evaluate four types of SFC embedding algorithms: nonlearning, NodeRank algorithms, benchmark algorithms, and reinforcement-learning-based algorithms.

Table 1. Simulation Parameter Setting.

Parameter	Values
Physical nodes	100
Physical links	600
CPU capacity	U[50,100]
Bandwidth capacity	U[50,100]
Link bandwidth resource	U[1,50]

5.2. Evaluation Results

During the testing phase, we evaluated our algorithm's performance across three key metrics using the SFCR from the test dataset. We conducted three sets of controlled trials focusing on long-term average revenue, the long-term SFCR acceptance rate, and the long-term average revenue–cost ratio, all while imposing time delay constraints on the algorithm. Furthermore, we compared our algorithm with the non-learning-based one, the baseline [29], and the SDSN algorithm [30]. The baseline algorithm adopts greedy node mapping and k-shortest path algorithm, prioritizing the mapping of virtual nodes to physical nodes with the maximum available resources, and selecting the shortest path of virtual links for mapping. The SDSN algorithm aims to achieve the minimization of total latency by selecting appropriate nodes and optimizing path and resource allocation while considering propagation delay and node processing delay.

5.2.1. Experiment 1: The Long-Term Average Revenue

This experiment evaluates algorithm performance by measuring the long-term average gain of the SFCR embedding as defined in Equation (2). Figure 2 illustrates the long-term average revenue of the four algorithms observed on a test network comprising 1000 SFCRs. Our algorithm achieved the highest revenue, surpassing the non-learning-based, baseline, and SDSN ones by 8.8%, 7.1%, and 13.7%, respectively. Initially, all four algorithms exhibited relatively high long-term average revenue, yet a decline was observed as training progressed. This decline can be attributed to the abundant network resources available at the outset of the test, which are capable of satisfying the constraints of most SFCR embeddings.

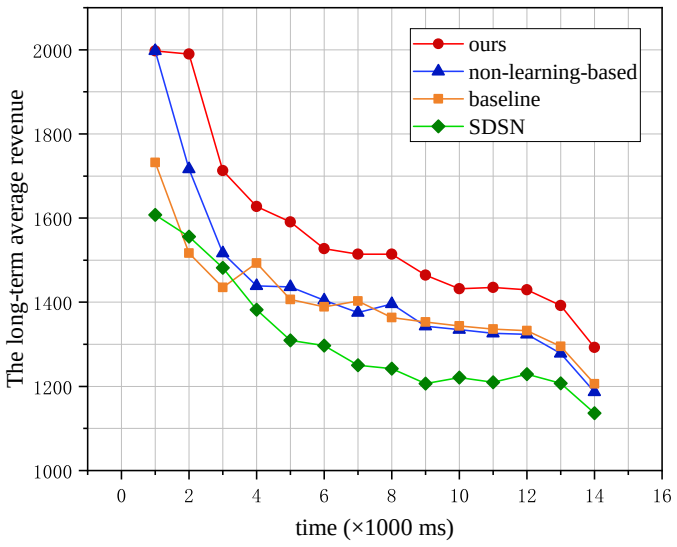

Figure 2. The long-term average revenue.

5.2.2. Experiment 2: The Long-Term SFCR Acceptance Rate

In the second test, we utilize Equation (1) as an evaluation metric. A high SFCR acceptance rate indicates the algorithm's robustness in handling SFCRs and its ability to generate substantial revenue. Figure 3 illustrates the evolution of the SFCR acceptance rate across the four algorithms, wherein the rate gradually declines throughout the experiment before stabilizing within a certain range. The SFCR acceptance rate of our algorithm exceeded that of the non-learning-based, baseline, and SDSN ones by 5.0%, 5.3%, and 4.7%, respectively.

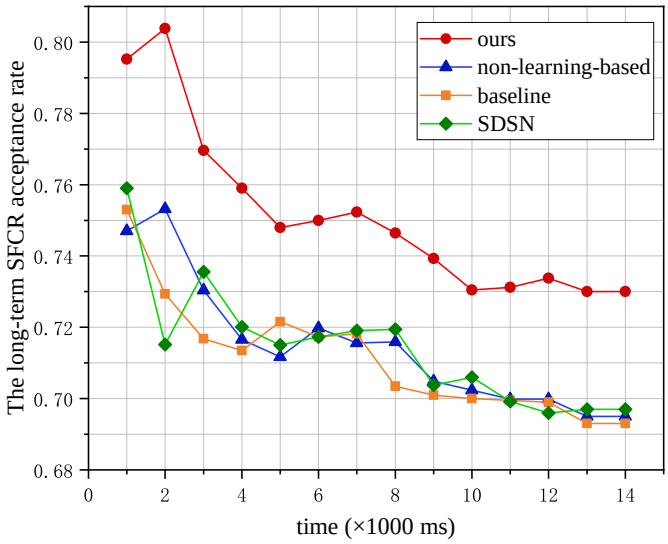

Figure 3. The long-term acceptance ratio.

5.2.3. Experiment 3: The Long-Term R/C Ratio

In the final experiment, we compare the long-term average revenue–cost ratios defined in Equation (3) across the four algorithms. Figure 4 illustrates the fluctuations and disparities in the long-term average revenue–cost ratio among these algorithms. The algorithm we proposed demonstrates metric values that are 2.7%, 2.7%, and 3.4% higher than those of the non-learning-based, baseline, and SDSN ones, respectively. Initially, when network resources are abundant, all four algorithms yield higher revenue–cost ratios. However, as the number of SFCRs increases, node and link loads escalate, leading to heightened network resource consumption and an inability to accommodate new SFCRs.

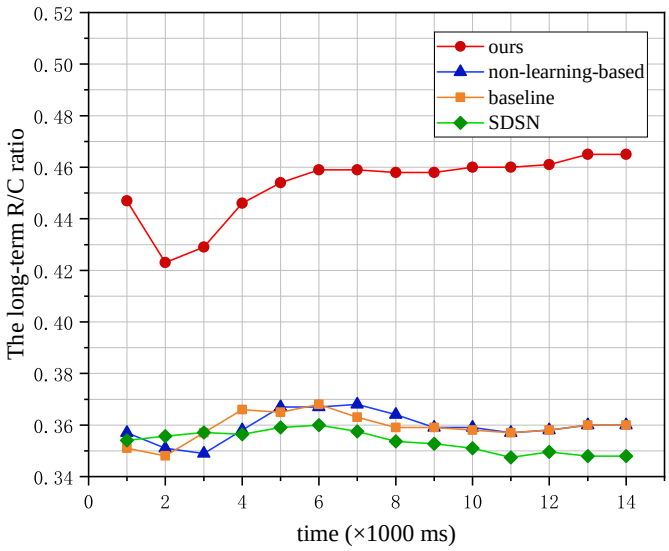

Figure 4. The long-term average revenue–cost ratio (R/C).

5.2.4. Summary of Evaluation

The three experiments we conducted demonstrate the effectiveness of the proposed reinforcement-learning-based SFC embedding algorithm in edge computing networks. When compared with traditional non-learning-based methods, baseline, and SDSN algorithms, the proposed algorithm exhibits significant advantages in terms of long-term average revenue, long-term SFCR acceptance rate, and long-term revenue–cost ratio. (1) For long-term average revenue, our algorithm consistently achieves the highest revenue, surpassing other methods. This can be attributed to the ability to dynamically adjust resource allocation based on the real-time state of the network, ensuring optimal resource utilization and maximizing revenue generation. (2) For the long-term SFCR acceptance rate, our algorithm has robustness in handling SFC requests and efficiently allocating resources to meet user demands. (3) For the long-term R/C ratio, our algorithm achieves a higher revenue–cost ratio, indicating better cost efficiency and profitability.

6. Conclusions

In order to cope with the challenges of resource allocation in edge computing networks, especially for the embedding requirements of business function chains, this paper delves into a reinforcement-learning-based resource allocation algorithm. By constructing a business function chain embedding model, the resource allocation problem is transformed into an optimization problem, and an efficient reinforcement learning algorithm is designed to solve it. The experimental results show that the algorithm proposed in this paper demonstrates obvious advantages in terms of throughput, delay, and fairness compared with traditional resource allocation methods. Meanwhile, in the process of this algorithm, we mainly focus on network performance and benefits. Therefore, we will incorporate resource collaboration, privacy security, and protection issues in edge computing networks into the enhanced learning framework, and we will aim to optimize our intelligent learning network and enhance the stability and large-scale network adaptability of the system in our future work.

Author Contributions: Conceptualization, W.W.; methodology, P.Z.; software, W.W.; formal analysis, P.Z.; investigation, P.Z. and K.L.; resources, S.C.; data curation, K.L.; writing—review and editing, S.C.; supervision, K.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work is partially supported by the Natural Science Foundation of Guangdong Province, China under Grant 2022A1515010999, partially supported by National Vocational Education Teacher Teaching Innovation Team Characteristic Project under Grant CXTD003, partially supported by the Shandong Provincial Natural Science Foundation under Grant ZR2023LZH017, ZR2022LZH015, partially supported by the Open Foundation of Key Laboratory of Computing Power Network and Information Security, Ministry of Education, Qilu University of Technology (Shandong Academy of Sciences) under Grant 2023ZD010, and partially supported by the National Natural Science Foundation of China under Grant 62341130, 62101300.

Data Availability Statement: The raw data supporting this article will be made available by the authors according to request.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Mahmood, O.A.; Abdellah, A.R.; Muthanna, A.; Koucheryavy, A. Distributed edge computing for resource allocation in smart cities based on the IoT. *Information* **2022**, *13*, 328. [CrossRef]
2. Cai, Q.; Zhou, Y.; Liu, L.; Qi, Y.; Pan, Z.; Zhang, H. Collaboration of heterogeneous edge computing paradigms: How to fill the gap between theory and practice. *IEEE Wirel. Commun.* **2023**, *31*, 110–117. [CrossRef]
3. Cao, H.; Wu, S.; Aujla, G.S.; Wang, Q.; Yang, L.; Zhu, H. Dynamic embedding and quality of service-driven adjustment for cloud networks. *IEEE Trans. Ind. Inform.* **2019**, *16*, 1406–1416. [CrossRef]
4. Bechtold, S.; Perrig, A. Accountability in future internet architectures. *Commun. ACM* **2014**, *57*, 21–23. [CrossRef]
5. Fisher, D. A look behind the future internet architectures efforts. *ACM SIGCOMM Comput. Commun. Rev.* **2014**, *44*, 45–49. [CrossRef]

6. Anderson, T.; Peterson, L.; Shenker, S.; Turner, J. Overcoming the Internet impasse through virtualization. *Computer* **2005**, *38*, 34–41. [CrossRef]
7. Houidi, I.; Louati, W.; Ameur, W.B.; Zeghlache, D. Virtual network provisioning across multiple substrate networks. *Comput. Netw.* **2011**, *55*, 1011–1023. [CrossRef]
8. Zhang, P.; Wang, C.; Qin, Z.; Cao, H. A multidomain virtual network embedding algorithm based on multiobjective optimization for Internet of Drones architecture in Industry 4.0. *Softw. Pract. Exp.* **2022**, *52*, 710–728. [CrossRef]
9. Cao, H.; Yang, L.; Zhu, H. Novel node-ranking approach and multiple topology attributes-based embedding algorithm for single-domain virtual network embedding. *IEEE Internet Things J.* **2017**, *5*, 108–120. [CrossRef]
10. Spano, S.; Cardarilli, G.C.; Di Nunzio, L.; Fazzolari, R.; Giardino, D.; Matta, M.; Nannarelli, A.; Re, M. An efficient hardware implementation of reinforcement learning: The q-learning algorithm. *IEEE Access* **2019**, *7*, 186340–186351. [CrossRef]
11. Wang, C.; Liu, L.; Jiang, C.; Wang, S.; Zhang, P.; Shen, S. Incorporating Distributed DRL Into Storage Resource Optimization of Space-Air-Ground Integrated Wireless Communication Network. *IEEE J. Sel. Top. Signal Process.* **2022**, *16*, 434–446. [CrossRef]
12. Abdullaev, I.; Prodanova, N.; Bhaskar, K.A.; Lydia, E.L.; Kadry, S.; Kim, J. Task offloading and resource allocation in iot based mobile edge computing using deep learning. *Comput. Mater. Contin.* **2023**, *76*, 1463–1477. [CrossRef]
13. Chen, Z.; Chen, Z.; Ren, Z.; Liang, L.; Wen, W.; Jia, Y. Joint optimization of task caching, computation offloading and resource allocation for mobile edge computing. *China Commun.* **2022**, *19*, 142–159. [CrossRef]
14. Liu, Q.; Li, J.; Wei, J.; Zhou, R.; Chai, Z.; Liu, S. Efficient multi-user for task offloading and server allocation in mobile edge computing systems. *China Commun.* **2022**, *19*, 226–238. [CrossRef]
15. Gao, J.X.; Hu, B.Y.; Liu, J.L.; Wang, H.C.; Huang, Q.Z.; Zhao, Y. Overbooking-Enabled Task Scheduling and Resource Allocation in Mobile Edge Computing Environments. *Intell. Autom. Soft Comput.* **2023**, *37*, 1–16. [CrossRef]
16. Yang, Y.; Gong, Y.; Wu, Y.C. Energy Sensitive Binary Offloading for reconfigurable-intelligent-surface-assisted wireless powered mobile edge computing. *IEEE Internet Things J.* **2023**, *11*, 11593–11605. [CrossRef]
17. Liu, X.; Yu, J.; Feng, Z.; Gao, Y. Multi-agent reinforcement learning for resource allocation in IoT networks with edge computing. *China Commun.* **2020**, *17*, 220–236. [CrossRef]
18. Wen, J.; Yang, J.; Wang, T.; Li, Y.; Lv, Z. Energy-efficient task allocation for reliable parallel computation of cluster-based wireless sensor network in edge computing. *Digit. Commun. Netw.* **2023**, *9*, 473–482. [CrossRef]
19. Li, Z.; Hu, C.; Wang, W.; Li, Y.; Wei, G. Joint access point selection and resource allocation in MEC-assisted network: A reinforcement learning based approach. *China Commun.* **2022**, *19*, 205–218. [CrossRef]
20. Zhi, Y.; Tian, J.; Deng, X.; Qiao, J.; Lu, D. Deep reinforcement learning-based resource allocation for D2D communications in heterogeneous cellular networks. *Digit. Commun. Netw.* **2022**, *8*, 834–842. [CrossRef]
21. Lin, Q. Dynamic resource allocation strategy in mobile edge cloud computing environment. *Mob. Inf. Syst.* **2021**, *2021*, 8381998. [CrossRef]
22. Li, F.; Fang, C.; Liu, M.; Li, N.; Sun, T. Intelligent Computation Offloading Mechanism with Content Cache in Mobile Edge Computing. *Electronics* **2023**, *12*, 1254. [CrossRef]
23. Xu, J.; Xu, Z.; Shi, B. Deep Reinforcement Learning Based Resource Allocation Strategy in Cloud-Edge Computing System. *Front. Bioeng. Biotechnol.* **2022**, *10*, 908056.
24. Sun, W.; Liu, J.; Yue, Y.; Wang, P. Joint resource allocation and incentive design for blockchain-based mobile edge computing. *IEEE Trans. Wirel. Commun.* **2020**, *19*, 6050–6064. [CrossRef]
25. Wang, C.; Liang, C.; Yu, F.R.; Chen, Q.; Tang, L. Computation offloading and resource allocation in wireless cellular networks with mobile edge computing. *IEEE Trans. Wirel. Commun.* **2017**, *16*, 4924–4938. [CrossRef]
26. Zheng, J.; Pan, Y.; Jiang, S.; Chen, Z.; Yan, F. A Federated Learning and Deep Q-Network based Cooperative Resource Allocation Algorithm for Multi-Level Services in Mobile Edge Computing Networks. *IEEE Trans. Cogn. Commun. Netw.* **2023**, *9*, 1734–1745. [CrossRef]
27. He, Y.; Wang, Y.; Lin, Q.; Li, J. Meta-hierarchical reinforcement learning (MHRL)-based dynamic resource allocation for dynamic vehicular networks. *IEEE Trans. Veh. Technol.* **2022**, *71*, 3495–3506. [CrossRef]
28. Zhang, P.; Li, Y.; Kumar, N.; Chen, N.; Hsu, C.-H.; Barnawi, A. Distributed Deep Reinforcement Learning Assisted Resource Allocation Algorithm for Space-Air-Ground Integrated Networks. *IEEE Trans. Netw. Serv. Manag.* **2023**, *20*, 3348–3358. [CrossRef]
29. Yu, M.; Yi, Y.; Rexford, J.; Chiang, M. Rethinking virtual network embedding: Substrate support for path splitting and migration. *ACM SIGCOMM Comput. Commun. Rev.* **2008**, *38*, 17–29. [CrossRef]
30. Cai, Y.; Wang, Y.; Zhong, X.; Li, W.; Qiu, X.; Guo, S. An approach to deploy service function chains in satellite networks. In Proceedings of the NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium, Taipei, Taiwan, 23–27 April 2018; pp. 1–7.