Computers, Materials & Continua

Tech Science Press

Doi:10.32604/cmc.2025.064654

ARTICLE

Enhanced Practical Byzantine Fault Tolerance for Service Function Chain Deployment: Advancing Big Data Intelligence in Control Systems

Peiying Zhang[1,2,*], Yihong Yu[1,2], Jing Liu[3], Chong Lv[1,2], Lizhuang Tan[4,5] and Yulin Zhang[6,7,8]

[1]Qingdao Institute of Software, College of Computer Science and Technology, China University of Petroleum (East China), Qingdao, 266580, China
[2]Shandong Key Laboratory of Intelligent Oil & Gas Industrial Software, Qingdao, 266580, China
[3]Library of Shanghai Lixin University of Accounting and Finance, Shanghai, 201209, China
[4]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, 250014, China
[5]Shandong Provincial Key Laboratory of Computing Power Internet and Service Computing, Shandong Fundamental Research Center for Computer Science, Jinan, 250014, China
[6]Key Laboratory of Ethnic Language Intelligent Analysis and Security Governance of MOE, Minzu University of China, Beijing, 100081, China
[7]Key Laboratory of Intelligent Game, Yangtze River Delta Research Institute of NPU, Taicang, 215400, China
[8]Key Laboratory of Education Informatization for Nationalities (Yunnan Normal University), Ministry of Education, Kunming, 650092, China

*Corresponding Author: Peiying Zhang. Email: zhangpeiying@upc.edu.cn

Received: 20 February 2025; Accepted: 03 April 2025; Published: 19 May 2025

ABSTRACT: As Internet of Things (IoT) technologies continue to evolve at an unprecedented pace, intelligent big data control and information systems have become critical enablers for organizational digital transformation, facilitating data-driven decision making, fostering innovation ecosystems, and maintaining operational stability. In this study, we propose an advanced deployment algorithm for Service Function Chaining (SFC) that leverages an enhanced Practical Byzantine Fault Tolerance (PBFT) mechanism. The main goal is to tackle the issues of security and resource efficiency in SFC implementation across diverse network settings. By integrating blockchain technology and Deep Reinforcement Learning (DRL), our algorithm not only optimizes resource utilization and quality of service but also ensures robust security during SFC deployment. Specifically, the enhanced PBFT consensus mechanism (VRPBFT) significantly reduces consensus latency and improves Byzantine node detection through the introduction of a Verifiable Random Function (VRF) and a node reputation grading model. Experimental results demonstrate that compared to traditional PBFT, the proposed VRPBFT algorithm reduces consensus latency by approximately 30% and decreases the proportion of Byzantine nodes by 40% after 100 rounds of consensus. Furthermore, the DRL-based SFC deployment algorithm (SDRL) exhibits rapid convergence during training, with improvements in long-term average revenue, request acceptance rate, and revenue/cost ratio of 17%, 14.49%, and 20.35%, respectively, over existing algorithms. Additionally, the CPU resource utilization of the SDRL algorithm reaches up to 42%, which is 27.96% higher than other algorithms. These findings indicate that the proposed algorithm substantially enhances resource utilization efficiency, service quality, and security in SFC deployment.

KEYWORDS: Big data; intelligent transformation; heterogeneous networks; service function chain; blockchain; deep reinforcement learning; trusted deployment

1 Introduction

With the swift evolution of Internet of Things (IoT) technology, device interconnectivity within heterogeneous network environments has markedly expanded. This expansion has facilitated diverse application scenarios such as intelligent homes and autonomous driving systems. Nevertheless, this highly interconnected environment also presents numerous security challenges [1], including breaches in data privacy, issues related to delegated trust, and risks during the deployment of Service Function Chaining (SFC) [2]. SFC is a technique that combines multiple network functions [3] in a specific sequence to cater to the unique requirements of various users. It offers tailored network services like traffic monitoring [4], firewall protection, and intrusion detection by organizing a series of virtual network functions (VNFs) [5] in a defined order. In 5G and IoT settings, SFCs allow for flexible configuration of network functions, enhancing efficient data processing and transmission [6]. However, dynamic challenges such as node failures, link congestion, and network attacks threaten the reliability of SFCs. Node failures can disrupt SFC operations, while malicious actors may tamper with requests, simulate nodes, or initiate denial-of-service (DoS) attacks [7], thereby jeopardizing network security, communication quality, and user experience.

Choosing optimal paths and nodes for deploying SFCs in intricate network topologies to meet quality of service (QoS) criteria, such as minimal latency, substantial bandwidth [8], and high reliability [9], constitutes a complex optimization challenge.

In response to these challenges, the present study introduces a trusted deployment algorithm for service function chaining based on an enhanced PBFT mechanism (VRPBFT) [10], aiming to optimize the traditional PBFT consensus mechanism by combining verifiable random function (VRF) and node reputation rating model. VRF is a cryptographic function that allows a node to generate a random number and verify it through its public key [11]. In VRPBFT, VRF is used for random number generation and verification. The algorithm integrates blockchain and deep reinforcement learning to construct a trustworthy SFC orchestration system. First, VRPBFT is designed, incorporating VRF and a node reputation model to reduce consensus latency and improve Byzantine node detection [12]. Second, a DRL-based SFC deployment algorithm (SDRL) optimizes deployment by dynamically adjusting node trustworthiness. The improved PBFT ensures consensus despite node failures or malicious behavior, enhancing system reliability. Experimental results show significant improvements in resource utilization, service quality, and security, outperforming existing algorithms in long-term revenue, request acceptance rate, revenue/cost ratio, and CPU utilization, offering an efficient and secure SFC deployment solution for heterogeneous networks [13].

2 System Model

This paper establishes a trusted network system centered on users, featuring heterogeneous physical nodes, blockchain, and deep reinforcement learning (DRL). DRL is a robust machine learning methodology that acquires decision-making capabilities through interaction with its environment, rendering it particularly suitable for dynamic and uncertain security contexts. DRL can enhance the performance of intrusion detection systems by facilitating adaptive learning, enabling dynamic decision-making, and minimizing false positives and false negatives. Blockchain and DRL form the system's core, with blockchain managing resource and transaction registration and updates during deployment [14]. The trusted network system is shown in Fig. 1.

Before service provisioning, information about all physical nodes is registered on the blockchain [15], ensuring secure VNF node deployment through secure data storage.

During SFC deployment, the blockchain module authenticates users and checks permissions while monitoring the resource and location data of physical nodes [16]. This information is passed to the DRL

optimization module to determine the best SFC deployment strategy [17]. The optimization steps are shown in Fig. 2.

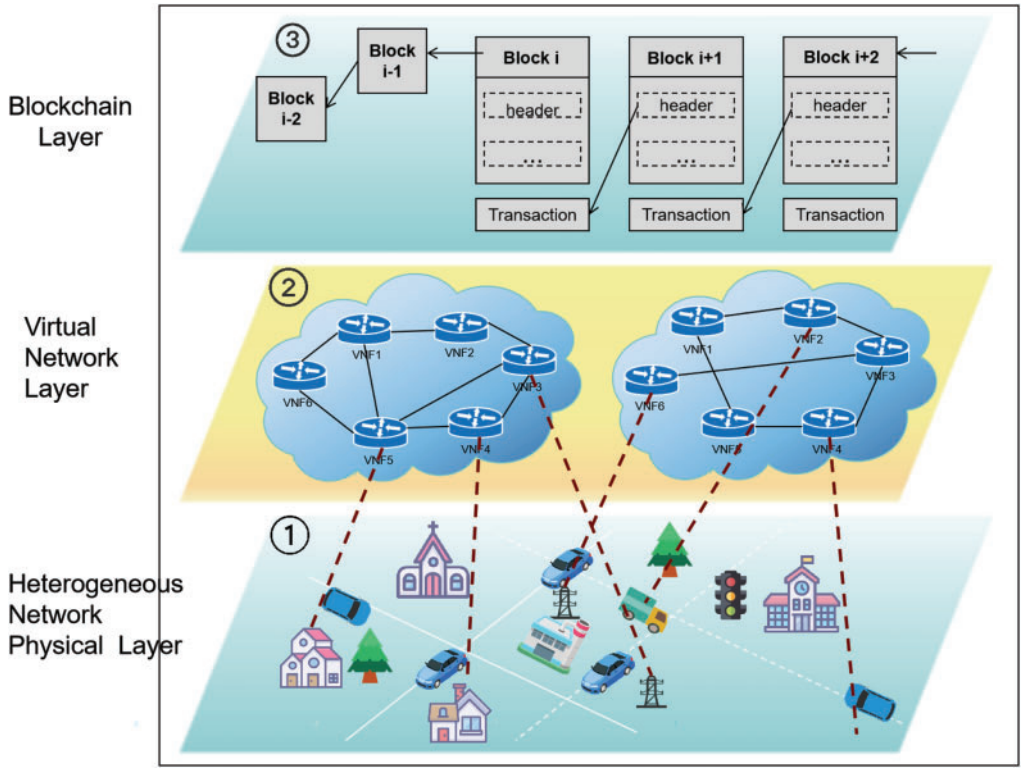

Figure 1: Trusted network system

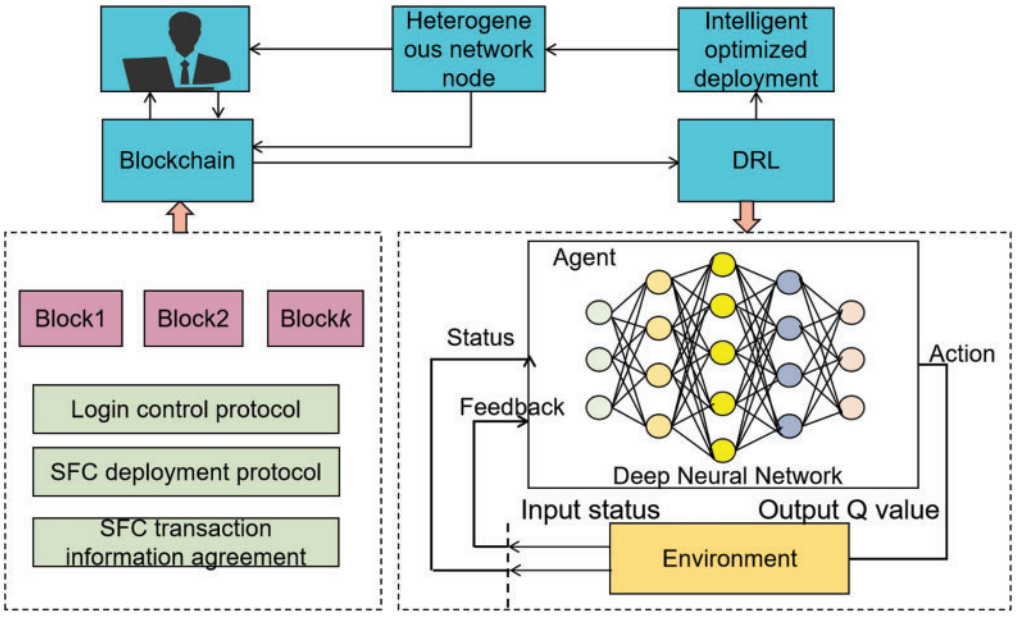

Figure 2: Working principle of trusted network layer

2.1 Network Model

The physical network model of the heterogeneous network is depicted as a weighted undirected graph $G^S = \{N^S, L^S, CPU(N_i^S), Eng(N_i^S), SL(N_i^S), BW(L_i^S)\}$. Here, $CPU(N_i^S)$ represents the CPU resources available at physical node n_i^S, $Eng(N_i^S)$ denotes the energy capacity of n_i^S, and $SL(N_i^S)$ indicates the security level of N_i^S. Additionally, $BW(L_i^S)$ specifies the bandwidth resources associated with physical link L_i^S.

The operational probability P of physical nodes and links adheres to a uniform distribution within the interval $[e^{-0.005(q-1)}, e^{-0.005q}]$.

2.2 Constraints

Service function chain deployment needs to follow certain constraints, including resource constraints for nodes and links and security level constraints. The specific constraint formulas are as follows:

$$sl_{N_i}^V < Tru_{N_m}^s$$

where $sl_{N_i}^V$ denotes the security level requirement of virtual node ${N_i}^V$ and $Tru_{N_m}^s$ denotes the trustworthiness of physical node ${N_m}^s$.

2.3 Algorithm Evaluation Metrics

The evaluation metrics comprise consensus delay, long-term average revenue, long-term average revenue-to-cost ratio, service function chain (SFC) request acceptance rate, and CPU resource utilization [18]. The consensus delay is calculated by the formula:

Evaluation metrics encompass consensus delay, long-term average revenue, long-term average revenue-to-cost ratio, SFC request acceptance rate, and CPU resource utilization [18]. Consensus delay is computed using the following formula:

$$\Delta t = t_{\text{confirm}} - t_{\text{generate}}$$

Benefits and costs are calculated as follows:

$$Rev_{VNR_{q,t,te}} = t_e \cdot \sum_{u=1}^{N_{V_q}} CPU_{N_u}^V + Eng_{N_u}^V \cdot sl_{N_u}^V + \sum_{i=1}^{L_{V_q}} b_{w_{r_i}^V} \tag{1}$$

$$Cost_{VNR_{q,t,te}} = t_e \cdot \sum_{u=1}^{N_{V_q}} \sum_{k=1}^{N_k^s} x_k^{(u)} \cdot CPU_{N_u}^V + Eng_{N_u}^V \cdot Tru_{N_u}^s + \sum_{i=1}^{L_{V_q}} \sum_{k=1}^{L_S} y_k^{(i)} \cdot BW_{L_i^V} \tag{2}$$

The long-term average benefit-cost ratio is calculated by the formula:

$$Rc(VNR_{q,t,t_e}) = \frac{\sum_{t=0}^{T-1} Rev_{VNR_{q,t,t_e}}}{\sum_{t=0}^{T-1} Cost_{VNR_{q,t,t_e}}}$$

3 Service Function Chain Trusted Deployment Algorithm

In order to optimize the security of heterogeneous networks, this paper introduces the Byzantine consensus mechanism (PBFT) of blockchain, and combines the verifiable random function (VRF) and the reputation hierarchy model with PBFT to design a more efficient and reliable consensus mechanism, VRPBFT. By integrating the reputation values of the nodes, the dynamic trustworthiness of the nodes is taken into account in extracting the attributes of the physical network.

3.1 Definition of Node Trustworthiness

The reputation of a node, denoted as $Tru^{s}_{N_i}$, serves as a comprehensive metric to evaluate the reputation value of node i during the T_{th} consensus round. This metric ranges within the interval [0, 1]. The symbols and their corresponding definitions employed in the node reputation calculation are presented in Table 1, with the associated formulas provided below:

$$Tru^{s}_{N_i} = \alpha_0 \cdot IN_i + \alpha_1 \cdot SUC_i + \alpha_2 \cdot HR_i + \alpha_3 \cdot HB_i$$

where IN_i denotes the input cost rate of the node, SUC_i denotes the communication success rate of the node, HR_i denotes the honesty rate of the node, and HB_i denotes the historical credibility. The parameter α represents the weights corresponding to each credibility assessment metric and the sum of all values is 1. The introduction of α_i balances the contribution of each indicator in node creditworthiness calculation, allowing flexible adjustment of their relative importance based on different application scenarios and demands, ensuring accurate assessment of node credit status.

Table 1: Symbols and their meanings used in node reputation calculation

Notation	Meaning
d_i	Deposit invested by node i
D	Total deposit
IN_i	Input cost ratio
N_i	Number of times node i successfully completed communications
N	Total number of times node i requested communication
SUC_i	Communication success rate
h_i	Number of times a node i honest behavior
H	Total participation count of node a in consensus processes
HR_i	Honesty rate
HB_i	Historical reputation value
CR^{T}_{i}	Reputation score of node i during the T_{th} consensus round

A node's credibility is a comprehensive evaluation of its state, performance, and dynamic behavior, determining its reliability and aiding network cooperation and decision-making. Nodes with high investment costs and communication success rates are highly credible, while those with low input costs or honesty rates are less credible. The node credibility evaluation algorithm is detailed in Algorithm 1.

Algorithm 1: Node reputation evaluation algorithm

Input: Let N represent the cardinality of the node set, and ON denote the historical node reputation registry.
Output: The new node reputation value record table is recorded as NN
1: Randomly initialize the policy network parameters;
2: Initialize the node reputation list;
3: **while** $i < Ndo$ **do**
4: nodeData=GetData(ON,i); //read data related to node i;

(Continued)

Algorithm 1 (continued)

5:

$$IN_i = \frac{d_i}{D} \in [0,1]; \quad \text{//input cost rate;} \tag{3}$$

6:

$$SUC_i = \frac{n_i}{N} \in [0,1]; \quad \text{//communication success rate;} \tag{4}$$

7:

$$HR_i = \frac{h_i}{H} \in [0,1]; \quad \text{//honesty rate;} \tag{5}$$

8:

$$Tru(N_i^S) = \alpha_0 \cdot IN_i + \alpha_1 \cdot SUC_i + \alpha_2 \cdot HR_i + \alpha_3 \cdot HB_i; \quad \text{//node reputation;} \tag{6}$$

9: **end while**
10: Update(ON); //Update the node reputation value record table;
11: NN=ON;
12: **return** NN;

3.2 Node Hierarchy and Transfer

The nodes are categorized into A, B, C and D by the node partitioning mechanism as shown in Fig. 3. Level A nodes, with the highest reputation value (β, 1.0], can become master nodes, participate in consensus, and verify random numbers, such as rigorously vetted banking servers in financial networks. Level B nodes, with a high reputation value (0.5, β), assist in consensus but lack full validation authority, representing partially vetted institutions. Level C nodes, with a medium reputation value (γ, 0.5], participate with limited authority, such as submitting but not validating transactions, often representing newly joined institutions requiring validation from Level A or B nodes. Level D nodes, with the lowest reputation value (0, γ), are excluded from consensus and require rigorous validation, typically representing unvetted entities. This hierarchy ensures network security and efficiency, incentivizing nodes to maintain good behavior. The β threshold separates high-reputation nodes from medium-reputation nodes, while γ separates medium-reputation nodes from low-reputation nodes, ensuring only reliable nodes participate in critical operations.

The classification of each node changes dynamically. Both class A and B nodes are eligible to serve as master nodes, whereas class C nodes can only participate in the consensus process. Class D nodes cannot participate in the consensus due to their low reputation value. This dynamically changing hierarchy helps to ensure secure, stable and efficient operation of the network. The consensus node set consists of three types of nodes, A, B, and C. The master node set, i.e., the honest node group. The master node set, i.e., the honest node group, contains two types of nodes, A and B. The authority of nodes of different classes is shown in Table 2.

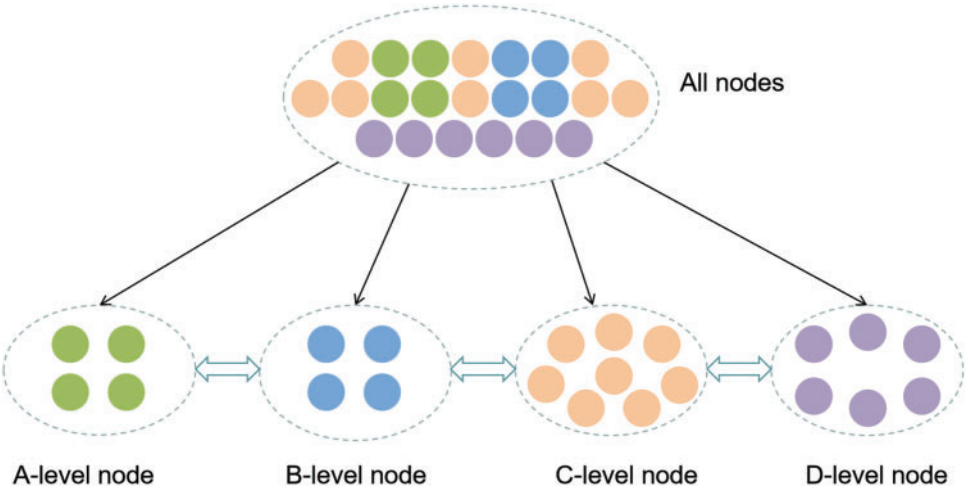

Figure 3: Schematic diagram of node classification

Table 2: Symbols and meanings used in node reputation calculation

Reputation level	Reputation value range	Priority master node	Master node	Consensus node
A	$(\beta, 1.0)$	✓	✓	✓
B	$(0.5, \beta]$	×	✓	✓
C	$(\gamma, 0.5]$	×	×	✓
D	$(0, \gamma)$	×	×	×

3.3 Master Node Selection

The nodes in class A and B are entitled to participate in the selection of master nodes to form an honest node cluster. The master node selection process comprises the following steps:

- Generation of VRF key pairs: each honest node generates a pair of VRF keys.
- Public Key Distribution: Nodes share their VRF public key for random number verification, keeping the private key secure.
- Random Number Generation: Nodes generate a random number using the VRF private key and current timestamp.
- Random Number Broadcast: Nodes broadcast the random number, VRF public key, and timestamp to the network.
- Random Number Verification: Other nodes validate the random number using the broadcasted VRF public key and timestamp.
- Random Number Sorting: The network collects and sorts all random numbers by size.
- Master Node Selection: The node with the smallest random number is designated as the master node. In case of a tie, the node with the highest reputation value is chosen.

3.4 SDRL Algorithm Design

The training procedure for the VRPBFT-based SFC deployment algorithm (SDRL) proposed in this paper is as shown in Algorithm 2.

Algorithm 2: VRPBFT-based SFC deployment algorithm

Require: physical network is recorded as G^S, virtual network is recorded as G^V, epoch = 200, */alpha* = 0.005, batch size =100;
Ensure: Probability that a node is embedded;
1: Randomly initialize the policy network parameters;
2: Initialize node reputation list;
3: **while** $counter < epoch$ **do**
4: $index = 0$;
5: **for** $SFCR \in trainingSet$ **do**
6: $M = extractFeatureMatrix(n_i)$; //extracting the feature matrix;
7: $p = getOutput(M)$; //Obtaining Deployment Probabilities;
8: $orOutput = VRF(n_i)$; //Generate pseudo-random values using VRF;
9: $filteredNodes = filterNodes(reputationList, orOutput, threshold)$; //Filter trusted nodes with reputation;
10: $host = select(filteredNodes, p)$; //Generate VNF embedding scheme in trusted nodes based on probability;
11: $loss = criterion(outputs, al)$; //Calculate loss;
12: getGradient(host); //Calculate gradient;
13: **if** isMapped($\forall VNF_i \in SFC$) **then**;
14: Links embedding; //Link embedding;
15: **end if**
16: **if** isMapped($\forall VNF_i \in G_V, \forall l_v \in G_V$) **then**
17: Calculate reward and gradients; //Calculate reward and gradient;
18: Backpropagation gradient;
19: Updating agent parameters; //Update intelligent body parameters;
20: UpdateReputation(reputationList, host, reward); //Adjust node reputation based on reward values;
21: **else**
22: Clear gradients; //Gradient clearing ;
23: **end if**
24: **end for**
25: Counter += 1;
26: **end while**

3.5 Link Deployment

In this study, the learning agent is responsible for determining the deployment strategy for each virtual network function (VNF). After completing the deployment of all VNFs, link deployment is realized through the shortest path algorithm [19]. This algorithm increases the redundancy and fault tolerance of the system and improves the stability of the virtual network by mining multiple shortest paths from the underlying physical network to meet the link connectivity requirements. The available resources in the underlying physical network are updated after the links are successfully deployed [20]. Algorithm 3 describes the specific steps of link deployment in detail.

Algorithm 3: Service function chain deployment algorithm

Require: *SFC* request G^V, physical network G^S, VNF scheme;
Ensure: SFC deployment scheme;
1: **for** $L_i^V \in L^V$ **do**
2: $E = \text{topK}(L_i^V, k = 5)$; //Derive the set of top k shortest paths;
3: Remove(E); //Remove paths that do not meet the constraints;
4: **if** $E \neq \varnothing$ **then**
5: **return** true;
6: **else**
7: **return** false;
8: **end if**
9: **end for**

4 Experiments and Analysis of Results

The proposed algorithm's effectiveness is demonstrated through comprehensive simulation analysis. Experimental findings reveal that the enhanced PBFT consensus mechanism achieves notable improvements in both consensus delay and security, effectively addressing Byzantine nodes and reducing the proportion of faulty nodes within the network. Moreover, the SDRL algorithm [21] outperforms baseline methods, with enhancements in long-term average revenue, request acceptance rate, revenue-to-cost ratio, and CPU resource utilization by 17%, 14.49%, 13.8%, and 27.96%, respectively. These results confirm that the proposed algorithm not only strengthens security but also optimizes network resource utilization efficiency, ensuring reliable and efficient SFC deployment under dynamic load conditions.

4.1 Experimental Environment Setup

A heterogeneous network topology comprising 100 physical nodes and 530 physical links was generated using the NetworkX tool [22] to simulate medium-sized ISP resources. Physical nodes are classified into X, Y, and Z categories, differing in CPU resources, memory resources, energy resources, bandwidth resources, and security levels. Concurrently, 2000 service function chain requests were generated, with 1000 allocated for training and 1000 for testing. The blockchain network was implemented via Hyperledger Fabric. Specific settings are outlined in Table 3.

Table 3: Parameter name and value range

Parameter name	Value range
Number of physical nodes	X nodes: 10 Y nodes: 25 Z nodes: 65
Physical Links	530
CPU resources of physical nodes	Node X: U [10, 30] cores Y nodes: U [20, 50] cores Z nodes: U [50, 100] cores
Security level of the physical node	U [A, B, C, D]

(Continued)

Table 3 (continued)

Parameter name	Value range
Bandwidth resources of physical links	X link: U [10, 30] Mbps Y-link: U [30, 50] Mbps Z link: U [50, 100] Mbps Inter-domain link: U [30, 60] Mbps

Additionally, two sets of 1000 Service Function Chain Requests (SFCRs) were generated using the same setup and utilized as training and test datasets. The number of nodes per SFCR is uniformly distributed between 2 and 10, and the probability of each node being directly connected to another node is 0.4. The arrival time of SFCRs follows a Poisson distribution, with an average of 4 SFCRs arriving every 100 time units. The demand for CPU resources in VNFs is uniformly distributed between 3 and 50 cores, and the demand for security level is distributed across [A, B, C, D]. Security level requirements are distributed between [A, B, C] and [D]. Attribute settings for service function chain requests are presented in Table 4.

Table 4: Parameter name and value range

Parameter name	Value range
Number of VNFs	U [2,10]
CPU Resource Requirement for VNF	U [3, 50] cores
Bandwidth Resource Requirement of SFC	U [3, 50] Mbps
Security attribute requirement for VNF	U [A, B, C, D]

4.2 Experimental Results

The VRPBFT algorithm exhibits reduced consensus latency when the number of consensus nodes remains constant, with this benefit becoming more significant as the number of nodes grows. Moreover, the algorithm excels in identifying and eliminating Byzantine nodes. After approximately 100 rounds of consensus, the proportion of faulty nodes within the network is substantially diminished, thereby enhancing the system's security.

As shown in Fig. 4, the VRPBFT algorithm exhibits lower consensus latency than traditional methods, with its efficiency advantage growing as the number of consensus nodes increases. This improvement stems from VRPBFT reducing the nodes involved in consensus, thereby decreasing overall delay. Additionally, VRPBFT significantly reduces Byzantine nodes after about 100 consensus rounds, outperforming PBFT in node selection and fault detection. By integrating VRF and node classification, VRPBFT enhances Byzantine node detection, isolation, and overall system security.

Similar to long-term average gains, the SDRL algorithm [21] achieves optimal performance in request acceptance rate, as shown in Figs. 4 and 5, outperforming other algorithms by up to 14.49%. However, physical network resources limit the number of service function chain requests that can be supported.

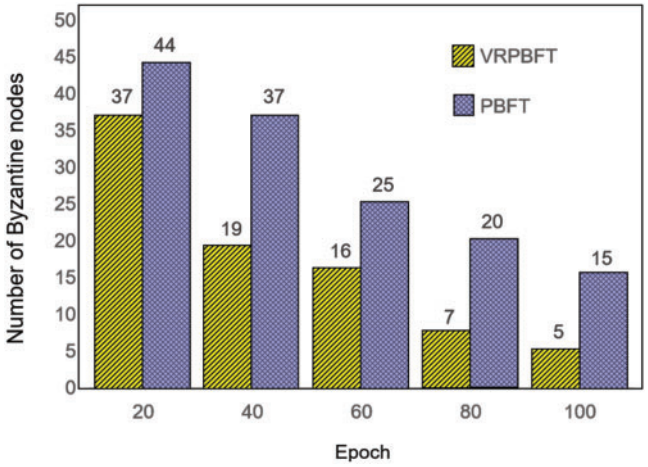

Figure 4: Consensus latency

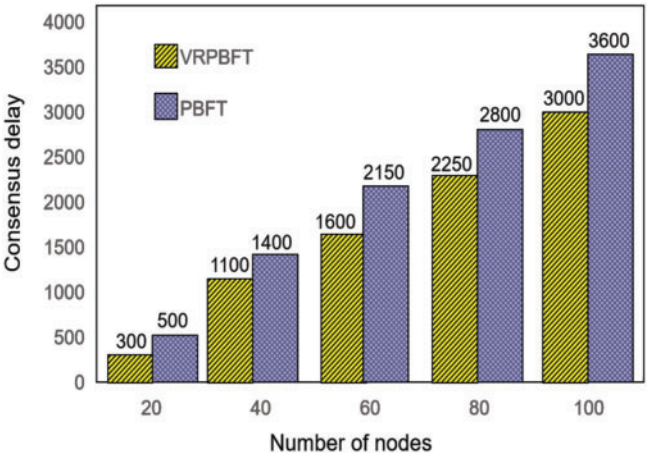

Figure 5: Comparison of Byzantine node counts

During training, the policy network's parameters undergo optimization starting from random initialization, as depicted in Fig. 6. This figure tracks four critical metrics for the SDRL algorithm: long-term average gain, gain-to-cost ratio, request acceptance rate, and CPU resource utilization [23]. These metrics increase over training iterations before eventually stabilizing [24]. In the early stages of training, agent performance is generally suboptimal due to the randomness of initial parameters [25]. However, performance improves progressively as training advances, reflecting enhanced algorithm effectiveness. Despite this, inherent limitations exist in performance improvement due to the algorithm's design. The SDRL algorithm demonstrates rapid convergence, improving the quality of SFC deployment [26]. To further validate the algorithm's efficacy, this chapter introduces the RL algorithm from [27] and the SA-RL algorithm from [28] as comparison benchmarks. The RL algorithm provides a decision-making framework, particularly suited for resource allocation and network optimization challenges. The SA-RL algorithm addresses both resource optimization and network security, aligning closely with the objectives of the SDRL algorithm proposed in this paper. Fluctuations in performance metrics during the training process of Deep Reinforcement Learning (DRL) are common. Firstly, the instability observed may stem from the algorithm's sensitivity to initial parameters, leading to an imbalance between exploration and exploitation. Secondly, insufficient training

iterations can prevent the algorithm from converging. Additionally, suboptimal hyperparameter selection can cause fluctuations in training. Dynamic changes in the environment necessitate continuous adaptation by the algorithm, which may also contribute to performance metric variability. Lastly, an inadequately designed reward function can result in unstable rewards in certain states, thereby affecting the learning process.

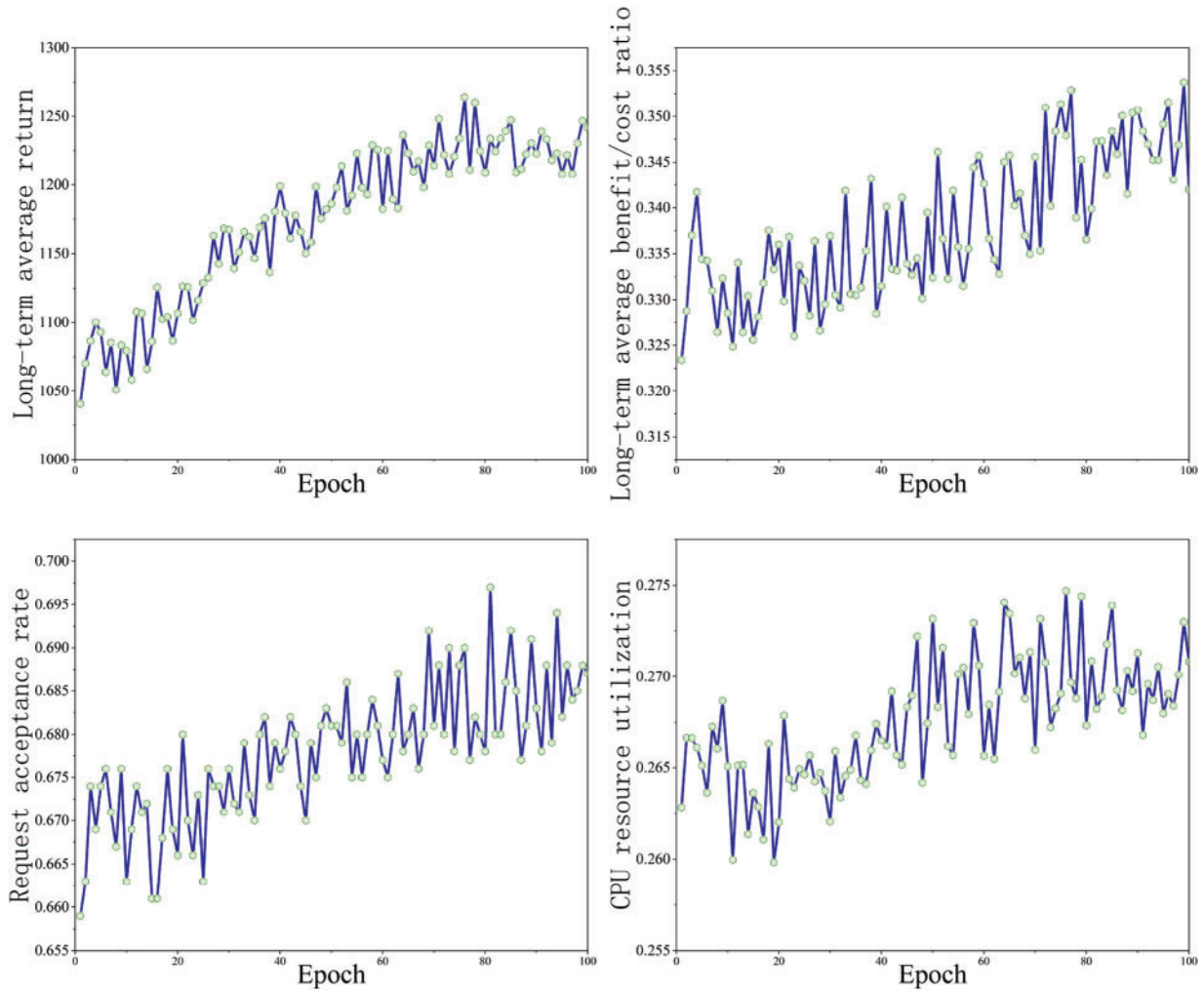

Figure 6: Changes of the four metrics throughout the training process of the SDRL algorithm

To address these challenges, we propose increasing the number of training iterations to ensure sufficient convergence. Hyperparameters will be optimized using methods such as grid search or random search to identify more suitable combinations. Algorithmic stability will be enhanced by employing more robust variants or incorporating regularization terms to mitigate overfitting. As can be seen in Fig. 7, the SDRL algorithm obtains the highest long-term average returns, followed by SA-RL. Compared to the other algorithms, The SDRL algorithm surpasses other algorithms by 17% and 19.86% in terms of long-term average returns.

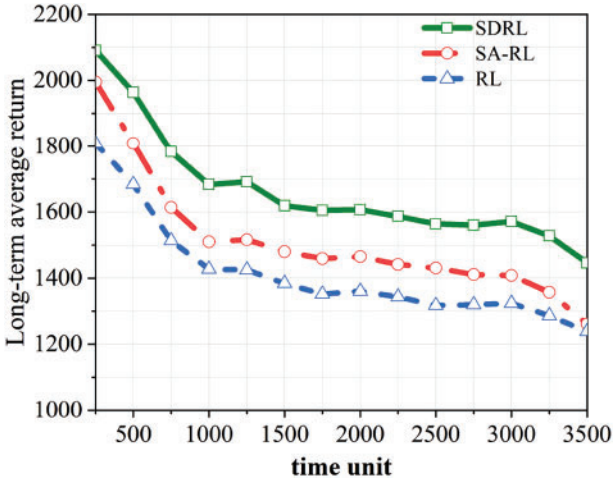

Figure 7: Performance evolution of distinct algorithms regarding sustained yield metrics

Therefore, both the long-term average gain and the service function chain request acceptance rate decay over time and eventually stabilize, as shown in Fig. 8.

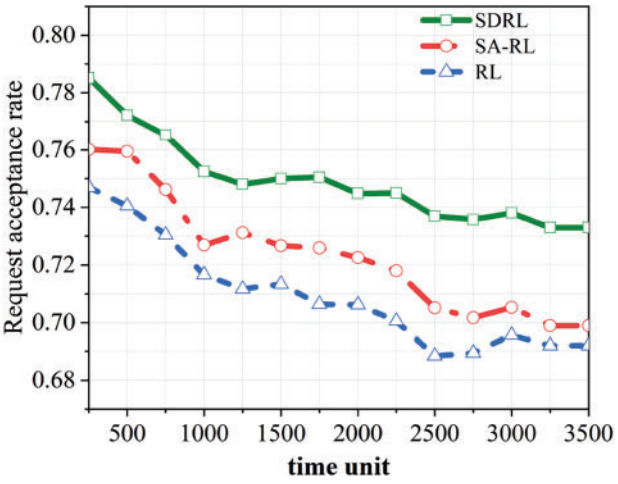

Figure 8: Comparative analysis of petition approval rates across computational methods

As illustrated in Fig. 9, the long-term average benefit-to-cost ratio follows a stable trend, reflecting the profitability of the algorithm. This metric depends on the deployment scheme's efficiency in using fewer physical links, independent of factors like network resources, security, and latency. The SDRL algorithm selects cost-effective deployment schemes, improving the benefit/cost ratio by 13.8% and 20.35% over the other algorithms.

In Fig. 10, the CPU resource utilization of SDRL ends up above 42%. The RL algorithm, based on reinforcement learning, neglects security constraints, while SA-RL considers node security but ignores dynamic trustworthiness changes, leading to resource wastage. These algorithms underperform due to insufficient consideration of constraints and dynamic changes. SDRL, by extracting dynamic trustworthiness and learning network attribute relationships, achieves more efficient deployment and higher resource

utilization. In summary, the algorithm proposed in this paper demonstrates greater stability. Compared to other algorithms, it achieves up to a 27.96% improvement in CPU resource utilization.

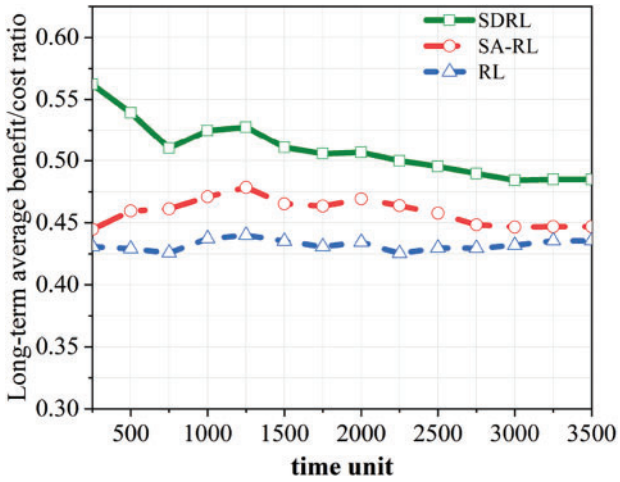

Figure 9: Comparison of changes in long-term average revenue/cost ratios for different algorithms

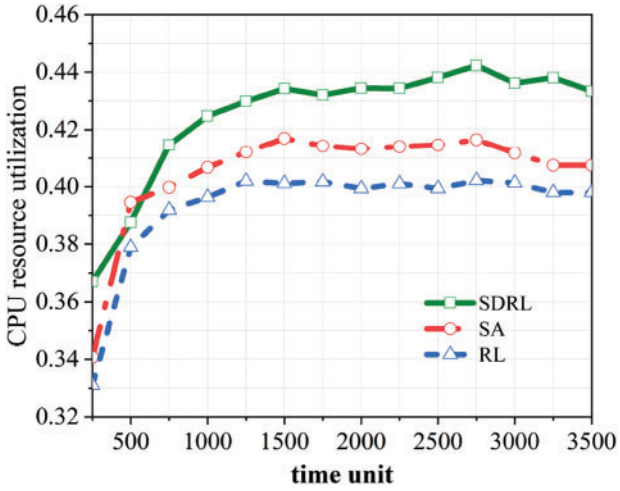

Figure 10: Comparison of changes in CPU resource utilization for different algorithms

Service function chain deployment scheme performance: The SDRL algorithm performs effectively across several metrics, including long-term average gain, request acceptance rate, benefit-to-cost ratio, and CPU resource utilization. Relative to comparison algorithms, it leads by 17%–19.86% in long-term average revenue, increases the request acceptance rate by up to 14.49%, enhances the benefit-to-cost ratio by 13.8%–20.35%, and achieves a final CPU resource utilization rate exceeding 42%, with an improvement of up to 27.96%.

5 Conclusions

This paper introduces VRPBFT, an enhanced PBFT consensus mechanism that integrates Verifiable Random Function (VRF) and a node reputation level model. This mechanism effectively reduces consensus delay and improves the efficiency of Byzantine node detection, decreasing the proportion of faulty nodes

by 40% after 100 rounds of consensus and reducing consensus delay by approximately 30% compared to traditional PBFT. Additionally, SDRL, a dynamic SFC deployment algorithm based on deep reinforcement learning, is designed to optimize resource allocation and enhance service quality through dynamic adjustments of node trust. It significantly outperforms existing algorithms in terms of long-term average revenue and other key performance indicators, such as a 17%–19.86% increase in long-term average revenue and a 14.49% improvement in request acceptance rate. The algorithm combines the reliability of blockchain with the trust of Byzantine nodes. The algorithm combines blockchain reliability and DRL intelligent decision-making capability, and excels in security, resource utilization and service quality, and can efficiently and reliably deploy SFCs under dynamic load, providing an effective solution for SFC deployment in heterogeneous networks. The algorithm has great advantages in enhancing security, optimizing resources, improving service quality and strong adaptability. Future work will focus on further optimizing the performance of the algorithm, expanding application scenarios, and enhancing security to cope with new security threats.

Acknowledgement: Thanks to the anonymous reviewers and editors for their hard work.

Funding Statement: This work is partially supported by the National Natural Science Foundation of China under Grant 62471493 and 62402257, partially supported by the Natural Science Foundation of Shandong Province under Grant ZR2023LZH017, ZR2024MF066 and 2023QF025, partially supported by the Open Research Subject of State Key Laboratory of Intelligent Game (No. ZBKF-24-12), partially supported by the Foundation of Key Laboratory of Education Informatization for Nationalities (Yunnan Normal University), the Ministry of Education (No. EIN2024C006), and partially supported by the Key Laboratory of Ethnic Language Intelligent Analysis and Security Governance of MOE (No. 202306).

Author Contributions: The authors confirm their contribution to the paper as follows: Conceptualization and Design: Peiying Zhang, Jing Liu, Chong Lv; Methodology: Peiying Zhang; Software: Peiying Zhang, Lizhuang Tan; Investigation: Yihong Yu; Data Curation: Yihong Yu, Lizhuang Tan; Funding Acquisition: Peiying Zhang, Jing Liu, Yulin Zhang; Project Administration: Peiying Zhang, Jing Liu; Writing—Original Draft: Yihong Yu, Peiying Zhang, Chong Lv; Writing—Review & Editing: Yihong Yu, Jing Liu; Supervision: Chong Lv, Yulin Zhang. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: The general created dataset is available upon request.

Ethics Approval: This study did not involve any human or animal subjects, and therefore, ethical approval was not required.

Conflicts of Interest: The authors declare no conflicts of interest to report regarding the present study.

References

1. Guo S, Dai Y, Xu S, Qiu X, Qi F. Trusted cloud-edge network resource management: DRL-driven service function chain orchestration for IoT. IEEE Internet Things J. 2019;7(7):6010–22. doi:10.1109/JIOT.2019.2951593.
2. Hantouti H, Benamar N, Taleb T. Service function chaining in 5G & beyond networks: challenges and open research issues. IEEE Netw. 2020;34(4):320–7. doi:10.1109/MNET.001.1900554.
3. Mao Y, Shang X, Yang Y. Joint resource management and flow scheduling for SFC deployment in hybrid edge-and-cloud network. In: IEEE INFOCOM 2022-IEEE Conference on Computer Communications; 2022; London, UK: IEEE. p. 170–9.
4. Qu H, Wang K, Zhao J. Reliable service function chain deployment method based on deep reinforcement learning. Sensors. 2021;21(8):2733. doi:10.3390/s21082733.
5. Torkzaban N, Baras JS. Trust-aware service function chain embedding: a path-based approach. In: 2020 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN); 2020; Leganes, Spain: IEEE. p. 31–6.

6. Wu TY, Wu H, Kumari S, Chen CM. An enhanced three-factor based authentication and key agreement protocol using PUF in IoMT. Peer Peer Netw Appl. 2025;18(2):83. doi:10.1007/s12083-024-01839-z.
7. Tsai TT, Lin HY, Huang WN, Kumar S, Agarwal K, Chen CM. Anomaly detection through outsourced revocable identity-based signcryption with equality test for sensitive data in consumer IoT environments. IEEE Trans Consum Electron. 2024;1–1:300.
8. Guo S, Qi Y, Jin Y, Li W, Qiu X, Meng L. Endogenous trusted DRL-based service function chain orchestration for IoT. IEEE Trans Comput. 2021;71(2):397–406. doi:10.1109/TC.2021.3051681.
9. Liu W, Zhang X, Feng W, Huang M, Xu Y. Optimization of PBFT algorithm based on QoS-aware trust service evaluation. Sensors. 2022;22(12):4590. doi:10.3390/s22124590.
10. Zhao H, Deng S, Liu Z, Xiang Z, Yin J, Dustdar S, et al. DPoS: decentralized, privacy-preserving, and low-complexity online slicing for multi-tenant networks. IEEE Trans Mobile Comput. 2021;21(12):4296–309. doi:10.1109/TMC.2021.3074934.
11. Luo J, Li J, Jiao L, Cai J. On the effective parallelization and near-optimal deployment of service function chains. IEEE Trans Parallel Distrib Syst. 2020;32(5):1238–55. doi:10.1109/TPDS.2020.3043768.
12. Li W, Feng C, Zhang L, Xu H, Cao B, Imran MA. A scalable multi-layer PBFT consensus for blockchain. IEEE Trans Parallel Distrib Syst. 2020;32(5):1146–60. doi:10.1109/TPDS.2020.3042392.
13. Manias DM, Shaer I, Naoum-Sawaya J, Shami A. Robust and reliable SFC placement in resource-constrained multi-tenant MEC-enabled networks. IEEE Trans Netw Serv Manag. 2024;21(4):187–99. doi:10.1109/TNSM.2023.3293027.
14. Santos GL, Endo PT, Lynn T, Sadok D, Kelner J. A reinforcement learning-based approach for availability-aware service function chain placement in large-scale networks. Fut Gener Comput Syst. 2022;136(1):93–109. doi:10.1016/j.future.2022.05.021.
15. Cao Y, Jia Z, Dong C, Wang Y, You J, Wu Q. SFC deployment in space-air-ground integrated networks based on matching game. In: IEEE INFOCOM 2023—IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS); 2023; New York, NY, USA: IEEE. p. 1–6.
16. Bhamare D, Jain R, Samaka M, Erbad A. A survey on service function chaining. J Netw Comput Appl. 2016;75(4):138–55. doi:10.1016/j.jnca.2016.09.001.
17. Zhu Y, Yao H, Mai T, He W, Zhang N, Guizani M. Multiagent reinforcement-learning-aided service function chain deployment for internet of things. IEEE Internet Things J. 2022;9(17):15674–84. doi:10.1109/JIOT.2022.3151134.
18. Chen J, Chen J, Zhang H. DRL-QOR: deep reinforcement learning-based QoS/QoE-aware adaptive online orchestration in NFV-enabled networks. IEEE Trans Netw Serv Manag. 2021;18(2):1758–74. doi:10.1109/TNSM.2021.3055494.
19. Sun G, Chen Z, Yu H, Du X, Guizani M. Online parallelized service function chain orchestration in data center networks. IEEE Access. 2019;7:100147–61. doi:10.1109/ACCESS.2019.2930295.
20. Khoshkholghi MA, Mahmoodi T. Edge intelligence for service function chain deployment in NFV-enabled networks. Comput Netw. 2022;219(2):109451. doi:10.1016/j.comnet.2022.109451.
21. Juan Y, Chaoqing X, Yize Z, Feifan S. Synchronous deep reinforcement learning (SDRL) algorithm for small batch image recognition. In: 2022 8th International Conference on Big Data and Information Analytics (BigDIA); 2022; Guiyang, China: IEEE. p. 317–23.
22. Meng W, Li W, Zhu L. Enhancing medical smartphone networks via blockchain-based trust management against insider attacks. IEEE Trans Eng Manag. 2019;67(4):1377–86. doi:10.1109/TEM.2019.2921736.
23. Liu Z, Xu X, Qiao P, Li D. Acceleration for deep reinforcement learning using parallel and distributed computing: a survey. ACM Comput Surv 2024;57(4):1–35.
24. Hu Y, Li T. Enabling efficient network service function chain deployment on heterogeneous server platform. In: 2018 IEEE International Symposium on High Performance Computer Architecture (HPCA); 2018; Vienna, Austria: IEEE. p. 27–39.
25. Sheng G, Min M, Xiao L, Liu S. Reinforcement learning-based control for unmanned aerial vehicles. J Commun Infor Netw. 2018;3(3):39–48.
26. Stooke A, Abbeel P. Accelerated methods for deep reinforcement learning. arXiv:180302811. 2018.

27. Yao H, Chen X, Li M, Zhang P, Wang L. A novel reinforcement learning algorithm for virtual network embedding. Neurocomputing. 2018;284(2):1–9. doi:10.1016/j.neucom.2018.01.025.
28. Zhang P, Wang C, Jiang C, Benslimane A. Security-aware virtual network embedding algorithm based on reinforcement learning. IEEE Trans Netw Sci Eng. 2020;8(2):1095–105. doi:10.1109/TNSE.2020.2995863.