EAI Endorsed Transactions
on Scalable Information Systems
Research Article EAI.EU

A Trusted Edge Scheduling Method for Computing Power Network Based on Deep Reinforcement Learning

Yong Yan[1], Ronghao Liu[1], Xu Liu[2,*], Ao Xiong[2] and Zuhao Wang[2]

[1]State Grid Zhejiang Electric Power Co., Ltd. Electric Power Science Research Institute, Hangzhou, China
[2]Beijing University of Posts and Telecommunications, Beijing, China

Abstract

In multi-party collaborative scenarios where computing power networks and edge intelligence work in synergy, trusted edge scheduling methods optimize resource allocation through intelligent scheduling, ensuring that computing tasks of each participant are executed on independent edge nodes, thus effectively improving computing efficiency and resource utilization. However, distributed edge nodes and their supporting communication networks usually require independent deployment and operation and maintenance, making efficient scheduling a critical challenge. This paper proposes a trusted edge scheduling method based on deep reinforcement learning, which realizes privacy protection through smart contract technology. Computing and network resources are stored on the blockchain, scheduling is performed using the Edge Routing Model (ERM) in smart contracts, and the hash values of scheduling results are recorded on the blockchain. ERM aims to meet users' computing demands while selecting the optimal routing strategy to maximize network bandwidth utilization. The method adopts deep reinforcement learning with a prioritized sampling mechanism to constrain the action space, and combines graph neural networks to learn features of network topology, thereby enhancing the generalization ability. Experimental results show that, compared with traditional scheduling models, ERM can schedule edge resources more efficiently under both homogeneous and heterogeneous topologies and exhibits favorable scalability.

Keywords: deep reinforcement learning, graph neural network, blockchain, smart contract, reverse proxy

Received on 18 May 2026, accepted on 23 September, published on 02 November 2026

Copyright © 2026 Yong Yan *et al.*, licensed to EAI. This is an open access article distributed under the terms of the CC BY-NC-SA 4.0, which permits copying, redistributing, remixing, transformation, and building upon the material in any medium so long as the original work is properly cited.

doi: 10.4108/eetsis.13053

1. Introduction

Multi-party collaboration driven by the integration of computing power networks and edge intelligence refers to the joint cooperation among multiple enterprises, institutions or departments to realize the sharing and utilization of cross-domain computing power and data. Trusted edge nodes provide a controlled computing environment that ensures data privacy and integrity while enabling secure data sharing and analysis among participants [1]. Trusted edge scheduling plays a critical role in multi-party collaborative scenarios. By coordinating and managing the allocation of edge computing resources, it optimizes resource utilization and effectively mitigates task conflicts and resource redundancy [2].

Conventional trusted edge scheduling methods typically adopt a resource management paradigm that separates computing resources from network resources. Most existing resource scheduling models follow this logic, treating computing and network resources as disjoint issues and focusing on solving specific subproblems. For computing resources, the main objectives include maximizing resource utilization, minimizing task completion time, and maximizing system performance. For instance, Kang Kaixuan et al. proposed an adaptive deep

*Corresponding author. Email: xuliu@bupt.edu.cn

reinforcement learning-based task scheduling framework for energy-efficient cloud computing, aiming to minimize task response time and maximize resource utilization [3]. For network resources, research focuses primarily on traffic routing selection. Sun Penghao et al. presented a scalable routing scheme based on deep reinforcement learning, which improves routing performance by adjusting link weights to alter routing paths between fixed nodes [4].

However, directly applying existing resource scheduling models to trusted edge scheduling may lead to several issues. When computing resources are considered in isolation, network congestion may occur during data transmission. Conversely, when only network resources are taken into account, edge nodes with the lowest network latency may become overloaded. Therefore, in trusted edge scheduling scenarios, treating computing resources and network resources separately fails to meet the requirements of overall coordinated resource optimization [5].

To address these challenges, it is particularly crucial to integrate computing resources and network resources into a unified platform for coordinated scheduling [6]. Therefore, this paper focuses on constructing a novel resource scheduling model for trusted edge scheduling, so as to realize the unified awareness, orchestration and coordination of computing power resources and network resources. This will help optimize resource allocation at edge nodes, improve network performance, and provide more flexible and efficient application services for all parties participating in multi-party collaboration of computing power networks.

Considering the complex service requirements of trusted edge scheduling in scenarios where computing power networks and edge intelligence are coordinated, as well as the characteristic that resource scheduling relies on powerful computing support and cannot be completed within a single network node, this paper adopts a centralized edge routing scheme, in which all scheduling decisions are uniformly formulated by the resource scheduling center. In the centralized edge routing scheme, the resource scheduling center can obtain real-time status information of trusted edge nodes and communication networks, and construct a global situation awareness map on this basis to guide the model in making accurate resource scheduling decisions. The objective of the centralized edge routing model is to achieve load balancing of global resources and improve overall resource utilization, thereby promoting the sustainable development of multi-party collaboration in computing power networks.

2. Related work

2.1. Reinforcement learning

Reinforcement learning is one of the main branches of machine learning [7]. It learns the optimal action strategy through the interaction process between the agent and the environment, and tries to maximize the current environment reward. As a more advanced reinforcement learning method, deep reinforcement learning uses deep neural networks to approximate Q-value function. Compared with traditional reinforcement learning, it can better apply to complex environments with high dimensional state space and action space. Therefore, deep reinforcement learning is often used to solve routing problems in networks [8].

2.2. Graph neural network

Graph neural network is a deep learning model specifically designed to process graph data [9]. In a graph neural network, each node represents an entity and each edge represents a relationship between entities. Graph neural network updates the representation of nodes gradually through the propagation and aggregation of information between nodes. After several iterations, graph neural network can capture the relative position and global structure information of nodes, extract rich node representation, and then perform tasks such as node classification, graph classification, and link prediction [10].

2.3. Routing protocol

Routing protocol is a set of rules that determines the best path for packets from the source node to the destination node, thus enabling efficient data transmission [11]. In order to improve the performance and reliability of the network, many studies are devoted to improving and optimizing the routing protocol. At present, the most common routing protocols include Routing Information Protocol (RIP), Open Shortest Path First (OSPF), and Border Gateway Protocol (BGP). These protocols use different algorithms and policies to determine the best path, taking into account factors such as bandwidth, latency, topology changes, among others. In general, research on routing protocols covers the design and implementation of new protocols, the improvement and optimization of existing protocols, as well as the research of security and robustness, which is of great significance for the construction of efficient, reliable and secure computer networks [12].

2.4. Reverse proxy

Reverse proxy is a technology specifically designed to process and forward client requests [13]. It acts as an intermediate layer between the client and the source server, receiving client requests and forwarding them to the source server, while returning the response to the client. Reverse proxy technology is mainly used in load balancing, security protection and cache optimization. In terms of load balancing, reverse proxies can distribute requests to multiple backend servers to achieve efficient resource utilization and even load distribution. To enhance security, reverse proxies can hide the real address of the source

server, prevent direct attacks, and implement access control. By caching static content, reverse proxies can also reduce the burden on the source server and improve response time. Common reverse proxy technologies include HAProxy and Nginx.

3. A trusted edge scheduling method based on deep reinforcement learning

3.1. Overview

In the scenario of collaboration between computing power networks and edge intelligence, the resource scheduling center receives user service requests and selects the optimal trusted computing power node and the optimal traffic routing path to that node from multiple distributed computing power nodes that meet user service requirements. It should be noted that this paper focuses on some traffic-critical services, which will not be released or replaced in the short term once identified. It can be seen that finding the optimal computing node and traffic routing for users is an NP-hard problem, and traditional resource scheduling models are no longer applicable. The emergence of deep reinforcement learning provides a new approach to solving such problems [14]. The deep reinforcement learning model can dynamically adjust the service scheduling strategy according to the current real-time status of computing power nodes and routing status, and explore the optimal resource scheduling scheme through continuous experience accumulation, so as to provide users with efficient and trusted services. Therefore, this paper proposes a trusted edge scheduling method based on deep reinforcement learning.

The main process of this method is as follows. A user sends a service request to the resource scheduling center. Upon receiving the request, the center forwards the request together with the current resource status to the Edge Routing Model (ERM) deployed on the smart contract. As a deep reinforcement learning model proposed in this paper, ERM can continuously adjust the scheduling policy according to the real-time status of computing nodes and link states, and derive the optimal resource scheduling scheme. Once ERM determines the optimal solution, the resource scheduling center schedules computing and network resources on the blockchain, computes the hash of the scheduling result, and records the hash value on the blockchain as evidence of the scheduling outcome.

In order to make the trusted edge scheduling method better applicable to scenarios involving the collaboration of computing power networks and edge intelligence, this paper makes two optimizations on the method:

First, as the number of nodes participating in multi-party data collaboration continues to grow in the computing power network, the number of distributed computing power nodes gradually increases. Multiple traffic routing paths exist between each computing power node and the request source, and the interaction between computing power nodes and routes generates a high-dimensional action space, making it difficult for the Edge Routing Model (ERM) to achieve satisfactory training performance. Therefore, it is necessary to design a reasonable computing power node selection algorithm to limit the number of candidate nodes, thereby reducing the training difficulty of the model.

Second, considering that the trusted edge scheduling method will face potential malicious requests and attack traffic when applied in the computing power network environment, it is necessary to design corresponding privacy protection strategies to prevent system crashes and information leakage.

To sum up, the scheduling process of the trusted edge scheduling method finally applicable to the collaborative scenario of computing power networks and edge intelligence is shown in Figure 1.

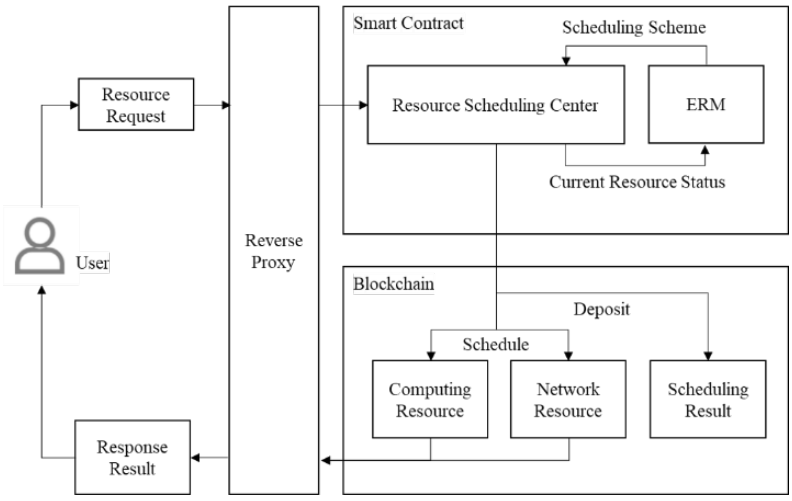


Figure 1. The process of trusted edge scheduling method

3.2. Edge routing model

For ERM, this paper needs to clarify the following definitions:

3.2.1 Environment state

In the trusted edge scheduling process based on deep reinforcement learning, ERM must comprehensively consider the computing resource and network resource requirements of the current service. In the resource scheduling scenario discussed in this paper, it is assumed that there exists at least one distributed computing power node in the environment that can satisfy the user's computing demands. The network state is defined by the characteristics of topological links, including link capacity, link centrality, the number of reachable paths, and the bandwidth required by the service. Specifically, link capacity denotes the available bandwidth on the current link; link centrality describes the number of traffic routing paths passing through the current link; the number of reachable paths equals the total number of candidate paths containing the current link; and the required bandwidth represents the link bandwidth resource to be allocated in the next step.

3.2.2 Action space

The action space is mainly determined by the number of candidate computing power nodes (CCN) and the number of candidate routes (CR) towards these nodes. Assume that there are N nodes to choose from, and each node reserves at most M candidate routes that meet service requirements. Then, each service request can generate at most N × M candidate routes to form the action space.

3.2.3 Agent and reward

ERM agents perform training primarily by interacting with the environment. In this process, the resource scheduling center obtains the environment status information from the global resource view. Based on the status information, the agent calculates the q value corresponding to the candidate route and selects the next action using a specific policy. If the action is successful, that is, there are traffic routes that meet service requirements, the ERM will receive a certain amount of reward.

The strategy ε-greedy, used to select the next action, can be expressed in (1), where ε represents the probability.

$$\pi(s|a) = \begin{cases} Select\ an\ action\ randomly, & \varepsilon \\ Select\ the\ optimal\ action, & 1-\varepsilon \end{cases} \quad (1)$$

This strategy is simple and effective, allowing for a balance between exploration and exploitation depending on the effectiveness of the training. In practice, ERM will directly select the action with the largest q value. In this paper, it is assumed that after the ERM executes the action successfully, that is, there is a traffic route that meets the user's service requirements, ERM immediately gets a reward proportional to the allocated resources; otherwise, the reward value is 0, and the calculation formula of the reward value is expressed in (2).

$$r = \begin{cases} \alpha \cdot c + \beta \cdot bw + \gamma \cdot s, & if\ successful \\ 0, & otherwise \end{cases} \quad (2)$$

In (2), α is the coefficient of computing resource c, β is the coefficient of bandwidth resource bw, and γ is the coefficient of storage resource s. By controlling the weight of different types of resources, the model can adapt to different business environments.

All interactive experiences generated during the iteration process are sequentially added to the experience replay pool for offline training. To address the problem that random sampling may cause the model to repeatedly learn low-value samples and slow down convergence, we introduce a prioritized sampling strategy in this work. The sampling probability of each experience sample is determined according to its TD error. Samples with significant prediction deviation will be sampled more frequently. In the subsequent model training phase, the agent updates network parameters by using the samples selected via prioritized sampling, which optimizes the learning performance and generalization ability of the deep reinforcement learning agent.

After defining the ERM, the scheduling process of ERM is shown in Algorithm 1.

Algorithm 1. The scheduling process of ERM

Input:

(1) Request source: src;

(2) Service computing power nodes set: dsts;

(3) Environment state: s;

(4) Reward value: reward;

(5) Number of computing power nodes: N;

(6) Number of routes: M;

(7) Request bandwidth resource: bw.

Output:

(1) Edge Routing Model (ERM)

1: Initialize the empty experience pool: exp_pool ← {};

2: Set the training termination flag: End ← False;

3: **While** not End **DO**

4: m_n_q_values ← {};

5: CCN ← Computing power node selection algorithm(N, src, dsts);

6: CR ← Routing selection algorithm(M, src, CCN);

7: **FOR** i in 0, … , M **DO**

8: **FOR** j in 0, … , N **DO**

9: p'← The j-th route of the i-th computing power node in CR;

10: s' ← Perform routing action, update state;

11: m_n_q_values[i][j] ← Calculate the q-value;

12: a ← Use ε-greedy to determine agent's action;

13: r,End,s',src',dsts',bw'← Perform action, get new states

14: Add the result of the round to exp_pool;

15: Update reward value: reward ← reward + r;

16: **IF** training_steps % K == 0: training model;

17: Update state: src ← src', dsts ← dsts', bw ← bw', s ← s'

3.3. Agent design

In deep reinforcement learning, an agent refers to an entity that learns how to take optimal action in a particular environment. The essence of ERM is the process of finding subgraphs in the network topology. If the graph neural network is introduced into the agent, the graph features of the network topology can be captured better and the generalization ability of ERM can be improved [15]. ERM's internal agent design is as follows:

When the agent receives the service request, it first obtains the candidate computing power nodes information and the link status information $l = \{x_1, x_2, x_3, x_4\}$ through the resource scheduling center. x_1 indicates the available bandwidth of the link, x_2 indicates the centrality of the link, x_3 indicates the number of reachable paths, and x_4 indicates the requested bandwidth of the service. v_{ik}^t indicates the number of connections that two links connect to the computing power node. Next, a message function M is constructed based on the model input to complete the state transmission, which is represented in (3).

$$M_k^{t+1} = \sum_{i \in N_k} (Concatenate(l_i^t + v_{ik}^t)) \quad (3)$$

N_k in (3) indicates the set of adjacent links. Concatenate aggregates the features of links and their neighbors to generate summary information. Then, the new hidden state of each link is calculated based on RNN [16].

$$l_k^{t+1} = RNN(l_k^t, M_k^{t+1}) \quad (4)$$

After the T-round iteration, the agent will generate the q value under the given state s and action a through the Readout function, which is represented in (5).

$$q = Readout\left(\sum_{k=1}^{K} l_k^T\right) \quad (5)$$

The Readout function in (5) consists of a fully connected neural network. In the edge routing model built based on DQN [17], the loss function is shown in (6).

$$L(\theta) = \frac{1}{N}\sum_j (y_j - q_j)^2 \quad (6)$$

In (6), θ represents the training network parameters, N represents the number of training samples, q_j represents the q value calculated by the agent for sample j, and y_j represents the target value, with the computation as shown in (7).

$$y_j = \begin{cases} r_j, & If\ step\ j\ is\ the\ last\ step \\ r_j + \gamma max_{a'} Q(s_{j+1}, a'; \theta^-), & otherwise \end{cases} \quad (7)$$

In (7), r_j represents the immediate reward obtained by performing action a_j under state s_j, γ represents the discount factor, and θ^- represents the target network model parameters.

3.4. Computing power node selection algorithm

The goal of computing power node selection algorithm is to limit the action space of the model and improve the convergence ability of the model. First, all the computing power node sets dsts that meet service requirements are filtered through the global view of resources. Then, the computing power node weights are calculated based on the computing power node selection algorithm. Based on the computing power node weights, no more than N computing power nodes are selected as CCN. The following three computing power node selection algorithms are proposed in this paper.

3.4.1 Based on average path length

Firstly, the average path length of all traffic routes meeting service requirements is calculated based on all computing power nodes in dsts. The path length refers to the number of hops between the request source point and the computing power node. The N computing power nodes with the shorter average path length are preferentially selected as the CCN.

3.4.2 Based on available resources

The resource scheduling center needs to monitor and evaluate the available resources of each computing power node in real time. The CCN preferentially selects N computing power nodes with the most available resources.

3.4.3 Shortest path first

For each computing power node in dsts, the shortest traffic route that meets user service requirements is reserved. Then, computing power nodes are sorted according to route length, and the top N computing power nodes are reserved as the CCN.

The above three computing power node selection algorithms are designed based on interpretability and implementation difficulty. When selecting a computing power node, it is necessary to consider the actual service scenario, and there is no optimal computing power node selection algorithm. Since the data that the three computing power node selection algorithms rely on already exists in the ERM implementation process, adding it to the ERM will have little impact on the execution efficiency of the model.

3.5. Privacy protection policy

The objective of the privacy protection policy is to hide the actual location and network structure of the computing power node, thereby reducing the risk of direct attacks on the computing power node by malicious users. The strategy realizes location hiding through reverse proxy technology [18]. The specific process can be described as follows:

First, an external user sends a service request to the reverse proxy server. At this point, the user does not know the specific location or server to which these requests will be forwarded, and this opacity effectively prevents the attacker from learning about the internal network topology.

Next, after receiving the user's request, the reverse proxy server will verify the validity of the request. This includes identity authentication, permission checking, and so on. At the same time, the reverse proxy server also checks the request content with strict security rules to ensure that there is no abnormal behavior. The reverse proxy server forwards the request to the internal resource scheduling center only after confirming that the request complies with the predefined security rules.

Then, after receiving the request from the reverse proxy server, the resource scheduling center sends the current resource status information and the request to ERM. ERM calculates the optimal resource scheduling scheme, generates the specific routing path, and sends the request to the appropriate computing power node.

After that, the computing power node receiving the request will parse, calculate, store and other processing operations on the requested data according to the predetermined operation logic. After performing these operations, the computing power node generates the corresponding response data, which is then returned to the reverse proxy server.

Finally, after receiving the response data from the computing power node, the reverse proxy server passes this data to the original requesting user.

Throughout the process, the user only interacts with the reverse proxy server and has no direct access to or recognition of the actual address of the internal computing power node. The privacy protection policy based on reverse proxy not only realizes the effective hiding of the actual location of the computing power node, but also prevents potential malicious attacks through request verification and content filtering. This privacy protection policy greatly improves the overall security and reliability of the trusted edge scheduling method.

4. Simulation and analysis

4.1. Preparatory phase

The ERM constructed in this paper is implemented based on TensorFlow framework, in which DQN model is selected for deep reinforcement learning, simulation scheduling environment is based on OpenAI Gym framework, NetworkX library and matplotlib library are used to build and display networks. Table 1 records the parameter configuration of the training process.

Topology data set from Topology Zoo was selected for the experimental phase, which included hundreds of real-world network topologies, such as Internet, data center, communication network, etc [19].

Table 1. Parameter Configuration

Parameter	Value
iterations	10000
Training episodes	10
Evaluation episodes	15
Learning rate	0.0002
Link state dimension	4
Batch size	50
Buffer size	200
gamma	0.99
α	1
β	0.1
γ	0.5
M	1

Since the introduction of GNN will complicate the model training, it is necessary to consider the influence of network topology on the model. Therefore, in the pre-processing stage, 107 network topologies in the Topology Zoo need to be classified, and two graph features of average link centrality and average node degree in the network topology need to be extracted. The average node degree is often related to the network topology density, and the higher the average node degree, the higher the network topology density. Link centrality describes the connectivity of a network. High link centrality means that paths are poorly distributed and multiple paths share the same link. The specific classification process is realized by clustering algorithm, and the experimental results are shown in Figure 2. Finally, after removing the special topology, the experiment obtained three clustering centers, namely AGIS, Deutsche Telekom and Amres, which can be used as the basic topology of the subsequent experiment.

Considering the scenarios where computing power networks and edge intelligence are coordinated, further configuration of resources is needed. First, link bandwidth resources are uniformly configured at 200 Mbps. Computing power node resources can be divided into computing resources and storage resources. The computing power node containing computing resources is configured with 16 vCPUs, and the computing power node containing storage resources is configured with 20TB. In addition, the user service request needs to be discretized, and the bandwidth resource request is in the unit of Mbps, including (8, 16, 32) three levels. The computing resource request is in vCPU unit, including (2, 4, 8) three levels. Storage resource requests are in terabytes and contain (0.5, 1, 2) three levels.

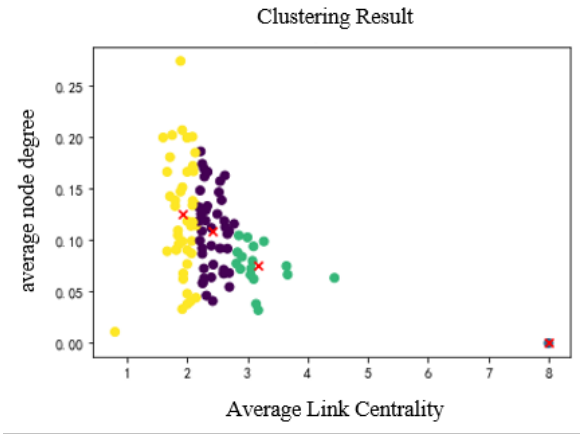


Figure 2. Clustering result

4.2. Contrast experiment

In order to reduce the action space dimension and avoid the difficulty of model training, it is necessary to use computing power node selection algorithm to limit the number of candidate computing power nodes. This section explores the impact of the number of candidate computing power nodes and the computing power node selection algorithm on ERM resource scheduling capability.

First, the basic network topology adopted for model training should be considered. Among the three candidate topologies obtained by clustering, AGIS has a large number of topologies, and the average link centrality and average node degree are both in an intermediate state, which is more general. Therefore, AGIS is chosen as the basic network topology environment, and the number of iteration rounds of model training is set to 2000. In terms of the number of candidate computing power nodes, the number of computing power nodes is set to increase from 1 to 6, and whether the model scheduling capability is positively correlated with the number of computing power nodes is explored. In terms of computing power node selection algorithm, four algorithms are set: average path length, available resources, shortest path first and random node selection. The experiment in this section is evaluated by score. The score represents the total reward value obtained by ERM for processing a set of business requests containing 100 requests. The higher the score, the more resource allocation ERM can accomplish.

The experimental results are shown in Figure 3. When the number of candidate computing power nodes is the same, the algorithm based on the average path length gets the highest score. In the case of the same algorithm, as the number of candidate computing power nodes increases, the score of most computing power node selection algorithms increases first and then decreases, and the score is the highest when the number of candidate computing power nodes is 3. This shows that the number of candidate computing power nodes is not positively related to the scheduling ability of the model, and too many candidate computing power nodes will lead to the difficulty of model convergence. Based on the experimental results, the computing power node selection algorithm based on the average path length will be used in subsequent experiments, and the number of candidate computing power nodes will be set to 3.

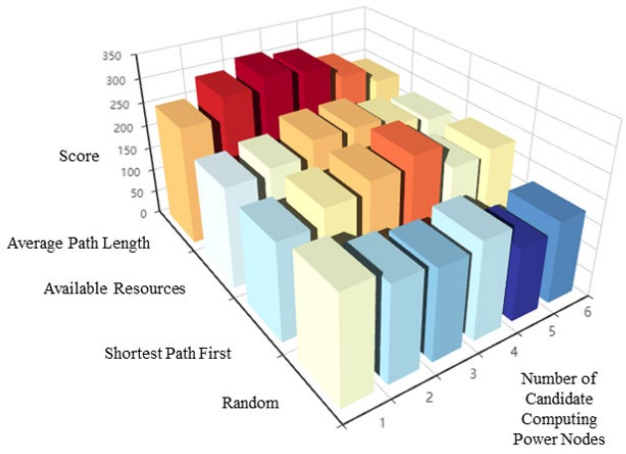


Figure 3. ERM training effect

4.3. Performance analysis

This section compares ERM with existing resource scheduling models, including BPPOLB, SF, and FLUID. Among them, BPPOLB will provide users with the most remaining resources for service, and use PPO algorithm to complete the routing of fixed request endpoint [20]. SF is to directly select the corresponding node of the shortest path that meets the business needs for service [21]. FLUID allows traffic to be allocated to different candidate paths in proportion to the available bandwidth of the link [22]. It should be noted that FLUID is only suitable for some networks with simple structures and divisible flows. It is considered as an ideal model, and cannot be used as a practical resource scheduling model.

The scores of the four models in the AGIS network topology are shown by the cumulative distribution function, as shown in Figure 4. According to the experimental results, the score of ERM is higher than that of BPPOLB and SF, which is closer to the FLUID model. This shows that ERM can accomplish more resource allocation than the existing resource scheduling model under the same resource request state.

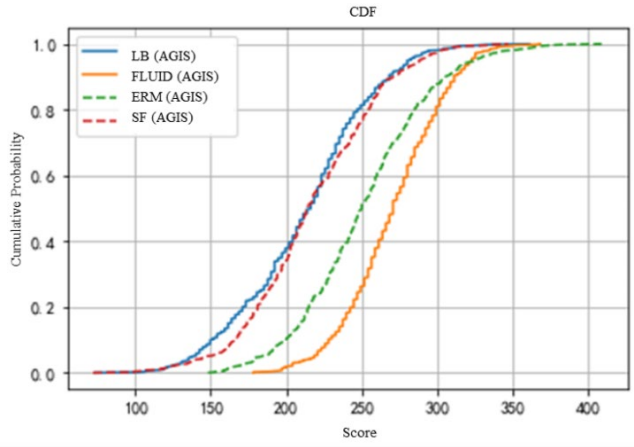


Figure 4. Scores of resource scheduling models based on AGIS

Figure 5 shows the scores of the four models in the Amres network topology, and the same experimental conclusions are obtained with the AGIS network topology. This shows that ERM is feasible and can be applied to heterogeneous network topologies.

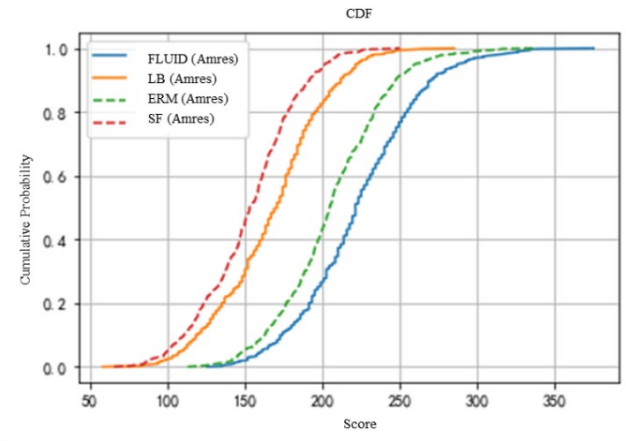


Figure 5. Scores of resource scheduling models based on Amres

4.4. Generalization ability

This section aims to test the generalization ability of resource scheduling model, explore whether ERM based on AGIS network topology training can achieve ideal results on similar network topology, and its ability to cope with link failures.

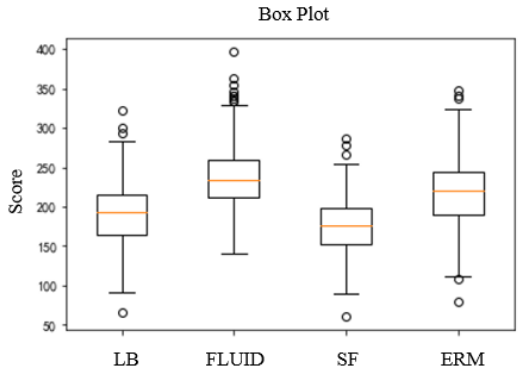


Figure 6. Scores of resource scheduling models based on Janet Backbone

The score of ERM in similar topologies is shown in Figure 6. Experimental results show that ERM still maintains good performance in Janet Backbone network topology, and its score is higher than that of BPPOLB and SF resource scheduling models and close to that of FLUID model, which proves that ERM has good generalization ability.

The ability of ERM to cope with link failures is shown in Figure 7. Experimental results show that the scores of each model decrease with the increase of the failure rate, but in most cases ERM can obtain relatively high scores and has a certain degree of fault tolerance.

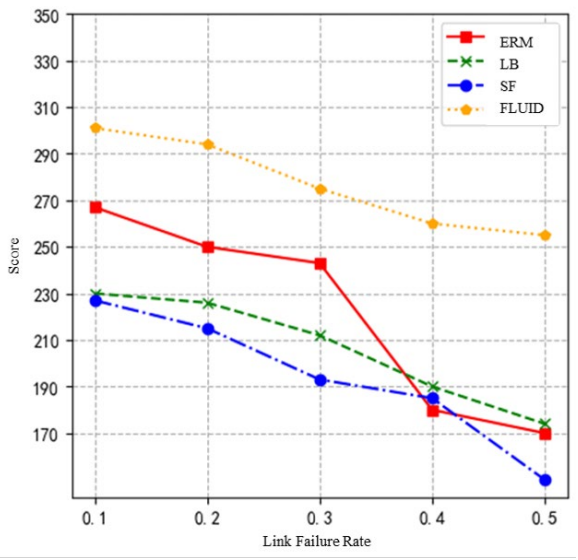


Figure 7. Scores of resource scheduling models under link failures

5. Summary

Compared with the traditional trusted edge scheduling method based on optimization algorithm, the trusted edge scheduling method in the scenarios where computing power networks and edge intelligence are coordinated pays more attention to the overall scheduling of computing resources and network resources. In the trusted edge scheduling method designed in this paper, a privacy protection policy based on reverse proxy is used to store computing resources, network resources and the hash values of scheduling results through blockchain, and ERM is deployed in smart contracts. Based on deep reinforcement learning technology, ERM provides computing and network resources to meet business requirements. ERM introduces graph neural network technology to improve its generalization ability, and also introduces an alternative computing power node selection algorithm to limit the size of action space and reduce the difficulty of model training. The experimental results show that ERM can complete more distributed computing tasks under the same service request and has good generalization ability compared with the existing edge scheduling method.

Considering that this paper does not split computing resource requirements and network resource requirements, we will further study how to split the two in the future, such as allocating computing requirements to multiple computing power nodes. This introduces a more complex environment that not only requires routing design work, but also requires consideration of more complex computing power node selection algorithms. In addition, the complexity of the environment will also have an impact on the construction and training of the model. The parallel training method similar to A3C distributed reinforcement learning can be considered, which can accelerate the learning process and improve the sample utilization rate.

Acknowledgements

This work was supported by State Grid Zhejiang Electric Power Co., Ltd. under Grant 5211DS23000D, as part of the science and technology project "Research and Application of Key Technologies for Multi-party Collaboration of Energy Data Based on Trusted Privacy Computing".

References

[1] Saleh A, Joshi P, Rathore R S, et al. Trust-aware routing mechanism through an edge node for IoT-enabled sensor networks. Sensors. 2022; 22(20):7820.

[2] Li J, Zhang H, Li S, et al. BD-TTS: A blockchain and DRL-based framework for trusted task scheduling in edge computing. Computer Networks. 2024; 251:110609.

[3] Kang K, Ding D, Xie H, et al. Adaptive DRL-based task scheduling for energy-efficient cloud computing. IEEE Transactions on Network and Service Management. 2022; 19(4):4948-4961.

[4] Sun P, Guo Z, Li J, et al. Enabling Scalable routing in software defined networks With deep reinforcement learning on critical nodes. IEEE/ACM Transactions on Networking. 2022; 30(2): 629-640.

[5] Avan A, Azim A, Mahmoud Q H. A state-of-the-art review of task scheduling for edge computing: A delay-sensitive application perspective. Electronics. 2023; 12(12): 2599.

[6] Tang X, Cao C, Wang Y, et al. Computing power network: The architecture of convergence of computing and networking towards 6G requirement. China Communications. 2021; 18(2):175-185.

[7] Moerland T M, Broekens J, Plaat A, et al. Model-based reinforcement learning: A survey. Foundations and Trends in Machine Learning. 2023; 16(1):1-118.

[8] Lyu L, Shen Y, Zhang S, The advance of reinforcement learning and deep reinforcement learning. In: 2022 IEEE International Conference on Electrical Engineering Big Data and Algorithms (EEBDA); Changchun, China. 2022. p. 644-648.

[9] Scarselli F, Gori M, Tsoi A C, et al. The graph neural network model. IEEE Transactions on Neural Networks. *2008;* 20(1):61-80.

[10] Zhou J, Cui G, Hu S, et al. Graph neural networks: A review of methods and applications. AI Open. 2020; 1:57-81.

[11] Masruroh S U, Fiade A, Iman M F, et al. Performance evaluation of routing protocol RIPv2, OSPF, EIGRP with BGP. In: 2017 International Conference on Innovative and Creative Information Technology (ICITech); Salatiga, Indonesia. 2017. p. 1-7.

[12] Rajeswari A R, Network A. Routing protocols: a comparative study. Recent trends in communication networks. 2020:225.

[13] Apache Software Foundation. Forward and Reverse Proxies. https://httpd.apache.org/docs/2.0/mod/mod_proxy.html, last accessed August 1, 2024.

[14] Kim M, Park J. Learning collaborative policies to solve np-hard routing problems. Advances in Neural Information Processing Systems. 2021; 34:10418-10430.

[15] Huang K, Zhai J, Zheng Z, et al. Understanding and bridging the gaps in current GNN performance

optimizations. In: Proceedings of the 26th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming. 2021. p. 119-132.
[16] Sherstinsky A. Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network. Physica D: Nonlinear Phenomena. 2020; 404:132306.
[17] Fan J, Wang Z, Xie Y, et al. A theoretical analysis of deep Q-learning. In: Learning for dynamics and control. PMLR. 2020. p. 486-489.
[18] Fraunholz D, Reti D, Duque Anton S, et al. Cloxy: A context-aware deception-as-a-service reverse proxy for web services. In: Proceedings of the 5th ACM workshop on moving target defense. 2018. p. 40-47.
[19] Knight S, Nguyen H, Falkner N, Bowden R, et al. The internet topology zoo. IEEE Journal on Selected Areas in Communications. 2011; 29(9):1765–1775.
[20] Garg S, Gupta S, Srivastava V. Resource allocation in iot networks via DQN and PPO algorithm. In: 2025 2nd International Conference on Computational Intelligence, Communication Technology and Networking (CICTN). 2025. p. 23-29.
[21] Anbazhagan K, Kumar R, Thilagavathy R, et al. Shortest job first with Gateway-based resource management strategy for fog enabled cloud computing. In: 2024 4th International Conference on Data Engineering and Communication Systems (ICDECS). 2024. p. 1-6.
[22] Alizadeh M, Yang S, Sharif M, et al. pFabric: Minimal near-optimal datacenter transport. In: Proceedings of ACM SIGCOMM. 2013. p. 435–446.