Computers, Materials & Continua

Tech Science Press

Doi:10.32604/cmc.2025.065665

ARTICLE

DRL-AMIR: Intelligent Flow Scheduling for Software-Defined Zero Trust Networks

Wenlong Ke[1,2,*], Zilong Li[1], Peiyu Chen[1], Benfeng Chen[1], Jinglin Lv[1], Qiang Wang[2], Ziyi Jia[2] and Shigen Shen[1]

[1]School of Information Engineering, Huzhou University, Huzhou, 313000, China
[2]National Key Laboratory of Advanced Communication Networks, Shijiazhuang, 050050, China

*Corresponding Author: Wenlong Ke. Email: wlke@zjhu.edu.cn

Received: 19 March 2025; Accepted: 14 May 2025; Published: 03 July 2025

ABSTRACT: Zero Trust Network (ZTN) enhances network security through strict authentication and access control. However, in the ZTN, optimizing flow control to improve the quality of service is still facing challenges. Software Defined Network (SDN) provides solutions through centralized control and dynamic resource allocation, but the existing scheduling methods based on Deep Reinforcement Learning (DRL) are insufficient in terms of convergence speed and dynamic optimization capability. To solve these problems, this paper proposes DRL-AMIR, which is an efficient flow scheduling method for software defined ZTN. This method constructs a flow scheduling optimization model that comprehensively considers service delay, bandwidth occupation, and path hops. Additionally, it balances the differentiated requirements of delay-critical K-flows, bandwidth-intensive D-flows, and background B-flows through adaptive weighting. The proposed framework employs a customized state space comprising node labels, link bandwidth, delay metrics, and path length. It incorporates an action space derived from node weights and a hybrid reward function that integrates both single-step and multi-step excitation mechanisms. Based on these components, a hierarchical architecture is designed, effectively integrating the data plane, control plane, and knowledge plane. In particular, the adaptive expert mechanism is introduced, which triggers the shortest path algorithm in the training process to accelerate convergence, reduce trial and error costs, and maintain stability. Experiments across diverse real-world network topologies demonstrate that DRL-AMIR achieves a 15–20% reduction in K-flow transmission delays, a 10–15% improvement in link bandwidth utilization compared to SPR, QoSR, and DRSIR, and a 30% faster convergence speed via adaptive expert mechanisms.

KEYWORDS: Zero trust network; software-defined networking; deep reinforcement learning; flow scheduling

1 Introduction

With the continuous development of emerging network applications such as the Internet of Things, artificial intelligence, and smart cities, traditional network perimeter-based security measures are increasingly ineffective. ZTN [1,2] enhances identity verification and fine-grained access control, enabling network systems to maintain high security and compliance in the face of increasingly complex threats. However, how to further enhance the flexibility of ZTN in flow control to improve the service quality remains an unresolved problem [3].

To enable more flexible management of network systems, SDN [4] has been progressively adopted. Currently, SDN is widely used in data center networks, local area networks, and wide area networks to

enhance service quality [5,6]. However, applying SDN to ZTN and designing corresponding network flow scheduling methods still presents challenges. First, ZTN requires more granular control over network traffic. Therefore, the designed flow scheduling methods must be capable of identifying different types of network flows. Second, the network flows of ZTN have varying security levels and transmission priorities. As a result, the flow scheduling methods must support differentiated flow control strategies [7,8].

To further improve the effectiveness and efficiency of flow scheduling methods in routing decisions, DRL [9,10] has been introduced. However, existing DRL-based flow scheduling methods still face the issue of slow model convergence, which hinders their ability to effectively respond to dynamic network changes [11]. To address these challenges, this paper proposes a flow scheduling method based on SDN and DRL for the ZTN scenario. The main contributions of this paper are as follows:

(1) We propose a traffic scheduling optimization model for ZTN. The proposed model takes into account the different requirements of various types of flow in ZTN for network performance and parameters such as delay, bandwidth, and path hop.
(2) We propose a flow scheduling method based on SDN and DRL, called DRL-AMIR. We tailor the state space, action space, reward function, and adaptive expert mechanism for DRL-AMIR to improve its performance.
(3) We conduct extensive testing of the proposed method in various network topologies. The experimental results show that the proposed method outperforms existing methods in terms of decision-making efficiency, transmission delay, load balancing, and bandwidth overhead.

The rest of this paper is organized as follows. Section 2 provides an analysis of the current research in the field. Section 3 introduces the traffic scheduling optimization model proposed in this paper. The DRL-AMIR method presented in this paper is described in Section 4. Section 5 outlines the experimental setup and results analysis. Section 6 concludes the paper.

2 Related Work

The core principle of zero trust network "never trust, always verify" is changing the traditional network boundary security model [12,13]. At present, zero trust technology has been widely used in network security, Internet of Things, cloud computing and other fields, showing the potential to solve security challenges in complex network environments [14]. Okegbile et al. [15] proposed a blockchain-enabled data sharing framework that enhances data security and credibility through distributed ledger technology in zero trust human digital twin systems. In their subsequent work on ZTN [16], they further improved the consensus process by enhancing system throughput and reducing latency through reputation mechanisms and sharding technology, which is crucial for the efficient operation of ZTN. Network traffic in ZTN can be classified according to security requirements and service characteristics [17]. However, existing research mainly focuses on device/user authentication and authorization, and the challenge of designing efficient routing optimization models for different traffic types in ZTN has not been largely resolved.

SDN realizes centralized control and flexible configuration of network management by separating control plane and data plane, which provides new possibilities for network flow scheduling [18]. Ali et al. [19] proposed an E2E QoS guarantee framework for IoT devices, achieving end-to-end quality of service guarantees in multi-domain heterogeneous networks through a two-layer SDN control architecture. Alenazi and Ali [20] developed a deep Q-learning scheme for QoS improvement in the physical layer of SDN, reducing end-to-end delay by intelligently selecting appropriate QoS categories. At present, SDN-based flow scheduling approaches demonstrate exceptional capabilities in optimizing latency, reducing energy consumption, and managing congestion control.

In recent years, DRL has been widely used in network routing optimization [21,22]. Shen et al. [23] proposed MFGD3QN, which integrates mean-field games with dueling double deep Q-networks to enhance network defense against attacks, providing insights for security-sensitive traffic optimization [24]. Casas-Velasco et al. [25] developed DRSIR, an intelligent routing method based on DRL and SDN, to generate adaptive routes for dynamic traffic changes. He et al. [26] proposed Message Passing Deep Reinforcement Learning (MPDRL), which uses graph neural network to interact with the network topology environment and extract knowledge through information transmission between links to achieve traffic load balancing [27]. Sun et al. [28] pointed out that existing DRL based routing solutions usually rely on complete node information. Their ScaleDeep method improves routing performance and enhances adaptability to topology changes by partially controlling key nodes in the network. In addition, research [29] shows that existing methods are often difficult to balance different performance indicators when considering multi-objective optimization, especially when dealing with traffic with strict deterministic requirements.

In general, the current DRL based network routing methods still have room for improvement in handling the differentiated routing requirements of multiple traffic types in the ZTN environment [30]. The DRL-AMIR method proposed in this paper aims to solve these limitations through state space, action space, reward function and adaptive expert mechanism specially designed for ZTN environment.

3 The Routing Problems in Software-Defined Zero Trust Networks

This section will analyze and model the routing problems in Software-Defined Zero Trust Networks. The goal of the modeling is to meet the routing requirements of different types of data flows in the ZTN, while considering the overall network load balancing.

3.1 Problem Statement

The core principle of ZTN is "never trust, always verify". In a ZTN, resource requesters can only establish a connection with the resource provider after identity authentication and authorization. This ensures the security of data resources and communication processes. At the same time, the ZTN can assign transmission priorities to network flows based on their identified flow types, ensuring that high-priority flow is transmitted over more suitable network links. This paper classifies the internal network flow in ZTN into three main categories based on its primary workflows:

(1) K-flow: These flows mainly involve network traffic related to system security, such as identity verification, authorization, and key distribution in a ZTN. This type of flows occupies a small portion of the network bandwidth but has the highest transmission priority.
(2) D-flow: These flows primarily involve network traffic generated when transmitting data related to specific user requests in a ZTN. This type of flows occupies a larger portion of the network bandwidth and has the second-highest transmission priority.
(3) B-flow: These flows mainly involve the transmission of background data within the ZTN. Since this data can be transmitted during system idle times, it requires a larger network bandwidth but has the lowest transmission priority.

The objective of routing and scheduling the aforementioned network flows in this paper is to minimize the transmission delay of high-priority flows and to achieve an overall load balancing across the network links. Furthermore, by routing the critical security-related traffic along the optimal transmission paths with minimized latency, the security performance of ZTN can be significantly enhanced.

3.2 Problem Formulation

The network topology of the ZTN can be represented as a graph $G(V, E)$, where V denotes the set of network nodes in the ZTN, and E represents the set of edges between the nodes. Based on the analysis of the overall ZTN business in Section 3.1, as well as the varying network performance requirements of different types of traffic during routing and transmission, this section focuses on the optimization and modeling of path selection for network flows during routing. The objective of this paper is to identify the transmission path that minimizes both the transmission delay and the number of hops, while also minimizing the maximum link bandwidth utilization. The following sections provide a detailed explanation of each objective.

(1) *Cumulative Hop Count*. The cumulative hop count refers to the total number of network links traversed by service flows as they are transmitted from the source node to the destination node. Reducing the cumulative hop count during data transmission can effectively minimize the overhead on network bandwidth and may also reduce transmission delays to some extent. Mathematically, the cumulative number of hops is defined as follows:

$$Chc\left(x_{f,e}\right) = \sum_{f \in F} \sum_{e \in E} x_{f,e} \tag{1}$$

where F denotes the set of all data flows in the network. Each flow $f \in F$ represents a distinct communication request between a source node and a destination node. e denotes a specific edge in the set E. $x_{f,e}$ denotes a binary decision variable that equals 1 if data flow f traverses edge e, and 0 otherwise.

(2) *Cumulative Transmission Delay*. The cumulative transmission delay refers to the total time taken for service flows to transmit from the source node to the destination node. It is the sum of the transmission delays of the network links that constitute the end-to-end transmission path. Minimizing the cumulative transmission delay helps improve the ZTN's response time to service requests, especially for delay-sensitive K-flows and D-flows, which have higher demands for transmission efficiency. The cumulative transmission delay is defined as follows:

$$Ctd\left(x_{f,e}\right) = \sum_{f \in F} \sum_{e \in E} d_e x_{f,e} \tag{2}$$

where d_e represents the transmission delay of edge e, which quantifies the delay required for data to traverse the link.

(3) *Maximum Link Utilization*. The maximum link utilization represents the highest bandwidth resource utilization among all links in the network. Minimizing the maximum link utilization effectively avoids bottleneck links, ensuring that traffic is evenly distributed across multiple links, thereby enhancing the load balancing performance of the network. Furthermore, this optimization objective improves the network's fault tolerance capability. When a specific link fails, the load on other links will not become excessively concentrated, preventing a sharp degradation in network performance. Mathematically, the maximum link utilization is defined as follows:

$$\mathrm{Mlu}\left(x_{f,e}\right) = max_{e \in E}\left(\frac{\sum_{f \in F} b_f x_{f,e}}{c_e}\right) \tag{3}$$

where b_f represents the bandwidth requirement of flow f. c_e denotes the bandwidth capacity of network link e.

(4) *Objective function*. The optimization objective of this paper is to minimize the three functions $Chc\left(x_{f,e}\right)$, $Ctd\left(x_{f,e}\right)$, and $\mathrm{Mlu}\left(x_{f,e}\right)$ during the routing selection process. The corresponding

objective functions are presented as follows:

$$min[Chc\left(x_{f,e}\right), Ctd\left(x_{f,e}\right), \text{Mlu}\left(x_{f,e}\right)] \tag{4}$$

Eq. (4) describes a multi-objective optimization problem. However, it is challenging to simultaneously achieve the minimum values for all optimization objectives. To reduce computational overhead, we transform the multi-objective optimization problem in Eq. (4) into a single-objective optimization problem with weights. This approach is justified as the Weighted Sum Method balances computational tractability with decision-maker priorities through adjustable coefficients, systematically guiding the search toward Pareto-optimal solutions. The corresponding single-objective functions are presented as follows:

$$min\ w_1 Chc\left(x_{f,e}\right) + w_2 Ctd\left(x_{f,e}\right) + w_3 \text{Mlu}\left(x_{f,e}\right) \tag{5}$$

$$\text{s.t.} \quad w_1 + w_2 + w_3 = 1 \tag{6}$$

$$\sum_{f \in F} b_f x_{f,e} \leq c_e \tag{7}$$

where w_1, w_2, and w_3 in constraint (6) are the weight coefficients for the three sub-objectives, and their values are typically determined through experiments or empirical knowledge. Constraint (7) indicates that for each link $e \in E$, the total traffic flowing through it must not exceed its bandwidth capacity.

To solve the routing optimization problem described by Eqs. (5)–(7), this paper proposes a flow scheduling algorithm based on deep reinforcement learning and an adaptive expert mechanism, termed DRL-AMIR. The specific details of this method will be introduced in the following section.

4 The Proposed DRL-AMIR Flow Scheduling Method

In this section, we first present the overall architecture of DRL-AMIR. According to existing research, the routing decision-making problem in networked systems constitutes a sequential decision-making task that can be modeled as a Markov Decision Process problem. Then, we provide a detailed explanation of DRL-AMIR's key modules, including the state function, action function, reward function, and adaptive expert mechanism.

4.1 The Overall Architecture

The framework of DRL-AMIR is designed as illustrated in Fig. 1, comprising three key components: the data plane, the control plane, and the knowledge plane. In the control plane, the controller communicates with the data plane via the OpenFlow protocol to achieve network topology awareness, link information measurement, and flow table distribution. The collected network link information is stored in the network link state repository within the knowledge plane, which is utilized by the DRL-AMIR agent during the training phase for learning. Once the agent training is completed, the controller constructs transmission paths based on the latest link information and distributes the corresponding routing flow tables.

4.2 The State Function

The state space comprises the following elements: node labels, link bandwidth and latency, length of the generated path, and type of service (ToS) field. Each element is utilized to represent different state information of the current network. The total length of the state vector is $|V| + 2|E| + 2$, where V denotes the

number of nodes in the network and E represents the set of edges between the nodes. The construction of the state vector is detailed as follows.

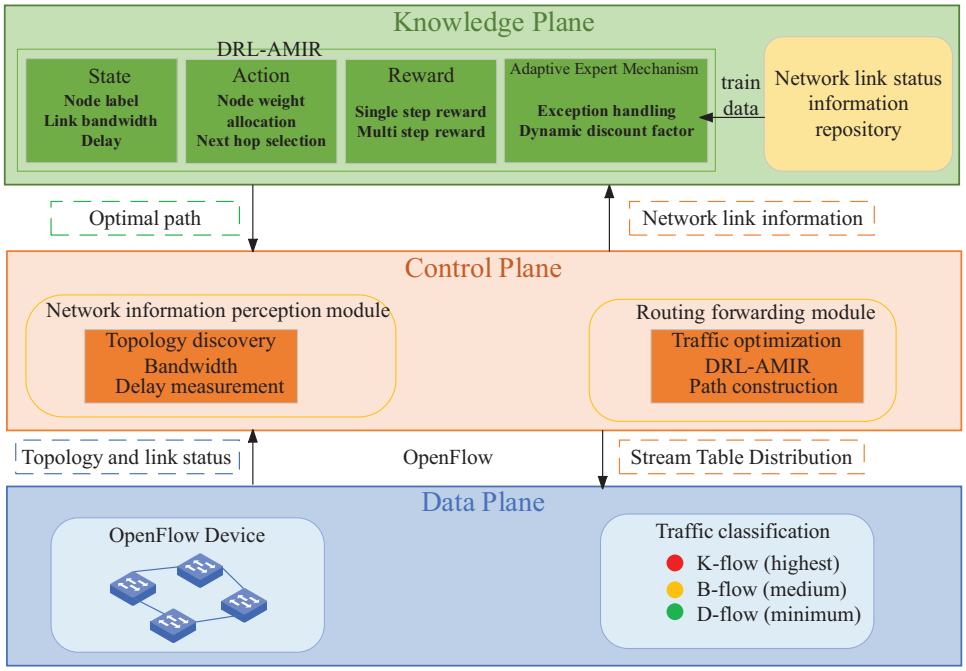

Figure 1: The framework of DRL-AMIR

A. Node labels

The node labeling section is employed to identify the source node, the destination node, and the nodes within the generated path. The vector length of node labels is $|V|$. For a routing path construction task, the node labeling process consists of the following three steps: (1) Initialize the first $|V|$ elements of the state space vector to 0. Each element corresponds to a node in the network, and its value is used to indicate whether the node has been selected for constructing the routing path. (2) Based on the requirements of the routing task, set the values of the vector elements corresponding to the source node and the destination node to n_s and n_d, respectively. (3) According to the agent's exploration in each round, sequentially label the nodes starting from the source node as n_i, where i represents the order of the node in the constructed routing path.

B. Link bandwidth and latency

In the state space, the link bandwidth and delay are used to record the remaining bandwidth and average transmission delay of each link in the current network, respectively. These data will directly influence the routing decisions of the DRL agent. When measuring the residual bandwidth of a specific link e_{v_1,v_2}, the SDN controller sends OFPPortStatsRequest messages to the two network nodes, v_1 and v_2, connected by this link. Upon receiving the request, each network node responds with an OFPPortStatsReply message, reporting the cumulative number of packets transmitted by the network port of the measured link. For nodes v_1, the reporting message is b_{v_1,t_1}. Then the residual bandwidth of v_1 can then be calculated as follows:

$$r_{v_1} = c_{v_1} - \frac{b_{v_1,t_1} - b_{v_1,t_0}}{t_1 - t_0} \tag{8}$$

where c_{v_1} represents the bandwidth capacity of the node v_1. b_{v_1,t_1} denotes the cumulative number of packets transmitted until time t_1, b_{v_1,t_0} represents the cumulative number of packets transmitted until the previous measurement time t_0, and $t_1 - t_0$ is the time interval between two consecutive measurements. Consequently, the residual bandwidth of link e_{v_1,v_2} at time t_1 can be determined, which is the smaller value between r_{v_1} and r_{v_2}.

To measure the transmission delay of link e, the SDN controller sends link layer discovery protocol (LLDP) packets to nodes v_1 and v_2, respectively, to determine the round-trip delay d_{v_c,v_1}, d_{v_c,v_2} between the controller and each node. Simultaneously, the total delay d_{v_c,v_1,v_2,v_c} is measured for a data packet traveling from the controller through nodes v_1 and v_2 and back to the controller. Based on these measurements, the transmission delay between nodes v_1 and v_2 can be calculated using the following equation:

$$d_{v_1,v_2} = \frac{d_{v_c,v_1,v_2,v_c} + d_{v_c,v_2,v_1,v_c} - d_{v_c,v_1} - d_{v_c,v_2}}{2} \tag{9}$$

where v_c represents the SDN controller. d_{v_c,v_2,v_1,v_c} denotes the transmission delay for a packet traveling from the controller through nodes v_2, v_1 and back to the controller.

C. Path length and ToS field

The path length refers to the length of the currently generated path, which is used to track the progress of the agent in exploring the complete path. In each round, the agent's action involves adding a link to the current path until a complete path from the source node to the destination node is constructed. The ToS field is utilized to indicate the type of traffic. It consists of an 8-bit binary value. In this study, the ToS fields for the K-flows, D-flows, and B-flows described in Section 3.1 are marked as 0b00000001, 0b00000010, and 0b00000011, respectively.

4.3 Action Function

The action space of the proposed DRL-AMIR method is designed as the set of weight values assigned to all nodes in the network. In each action step, the agent selects the next-hop node for the transmission path based on these weight values, continuing until a complete path from the source node to the destination node is constructed. Mathematically, the action space is represented as follows:

$$a_t = \left[w_{v_1}, w_{v_2}, \ldots w_{v_i}, \ldots, w_{v_n}\right] \tag{10}$$

where w_{v_i} denotes the weight assigned by the agent to the i-th node as the next-hop node. A higher weight value indicates a stronger preference of the agent for selecting that node.

Based on the current network state, the agent may encounter four distinct scenarios during action selection and environment interaction. Each corresponding to a specific reward or penalty strategy to guide the learning process.

(1) Valid Hop Reward: If the agent selects a next-hop node that is reachable, the state is updated to the new network state, and the agent receives a positive reward. This reward mechanism encourages the agent to choose valid next-hop nodes, thereby progressively advancing the path construction.
(2) Invalid Selection Penalty: If the agent selects a node that is unreachable, the current state remains unchanged, and the agent is penalized for the invalid selection. This penalty mechanism aims to reduce the agent's tendency to choose invalid nodes, promoting the selection of nodes that contribute to valid path construction.
(3) Loopback Penalty: When the agent selects a node that already exists in the current path, it incurs a loopback penalty. This penalty strategy prevents the inclusion of duplicate nodes in the path, avoiding

the formation of invalid or redundant hops, thereby enhancing the efficiency and effectiveness of path selection.

(4) Cycle Penalty: If the agent's selected node results in the formation of a cycle, it is penalized with a cycle penalty. This design prevents the creation of closed loops in the path, ensuring that network traffic does not circulate indefinitely and guaranteeing successful data delivery to the destination node.

Through the aforementioned action space design and reward mechanisms, the agent can progressively learn the optimal path selection strategy. It effectively avoids invalid path choices, cycles, and redundant node hops, thereby achieving efficient and high-quality data transmission.

4.4 Reward Function

In the proposed DRL-AMIR routing method, the reward function is designed to evaluate each action taken by the agent during the path selection process. The reward is used to guide the agent to learn in a direction that continuously optimizes network performance. The primary objectives of the reward function are to minimize path length, minimize transmission delay, and minimize maximum link utilization. To address the dimensional differences among these performance metrics, this study employs the range normalization method to standardize these indicators. The normalization process is as follows:

$$u(x) = \frac{x^{max} - x}{x^{max} - x^{min}} \tag{11}$$

According to Eq. (11), the smaller the value of a network performance metric x, the larger its corresponding normalized value $u(x)$. For any network performance metric x, the range of $u(x)$ is $[0, 1]$. To effectively guide the agent in finding the optimal transmission path, the reward function of DRL-AMIR includes both single-step rewards and multi-step rewards. Their mathematical formulations are presented below:

$$R_{step} = \lambda_1 u\left(Ctd(v_i)\right) + \lambda_2 u\left(Mlu(v_i)\right) \tag{12}$$

$$R_{finish} = w_1 u\left(Chc(p_{v_s,v_d})\right) + w_2 u\left(Ctd(p_{v_s,v_d})\right) + w_3 u\left(Mlu(p_{v_s,v_d})\right) \tag{13}$$

where $Ctd(v_i)$ and $Mlu(v_i)$ represent the scores in terms of transmission delay and link bandwidth utilization, respectively, after adding node v_i. Meanwhile, $Chc(p_{v_s,v_d})$, and $Mlu(p_{v_s,v_d})$ denote the total scores in terms of hop count, transmission delay, and link bandwidth utilization, respectively, after constructing the complete path p_{v_s,v_d} from the source node v_s to the destination node v_d. The parameters λ_1, λ_2, w_1, w_2, and w_3 are weighting factors. By dynamically adjusting the weighting factors in the reward function, the proposed method can adapt to the transmission requirements of different traffic flows. This enables the reduction of transmission delays for high-priority flows while simultaneously minimizing overall network bandwidth consumption and improving load balancing performance.

4.5 Adaptive Expert Mechanism

An adaptive expert mechanism is tailored for DRL-AMIR to further reduce the trial-and-error cost of the agent and improve training efficiency. The expert mechanism is triggered when the agent's decision-making meets any of the following conditions: (1) the selected node is unreachable; (2) the agent repeatedly selects a node already present in the current path; (3) the selected node would form a loop.

Once the adaptive expert mechanism is triggered, the agent employs a shortest-path algorithm to assist in selecting the next-hop node. The actions chosen by the expert are stored in the experience replay buffer with a discount factor D, ensuring that these actions appropriately influence the training process without

excessively interfering with the agent's autonomous exploration. The discount factor D dynamically adjusts based on the number of error rounds, following the equation:

$$D_{T+1} = min\left(1, max\left(\epsilon, D_T + k \cdot \left(\frac{M}{2} - T\right)\right)\right) \tag{14}$$

where ϵ denotes the minimum value of the discount factor. D_T represents the current discount factor. k is the step size coefficient controlling the adjustment speed of the discount factor. M represents the maximum number of rounds for the expert mechanism, and T denotes the current expert round. Through this adaptive expert mechanism, the agent can rapidly acquire effective path experiences during the early stages of training, thereby reducing unnecessary trial and error. Additionally, it ensures that the agent maintains a higher level of exploration during the mid-training phase. The process of the proposed DRL-AMIR is detailed in Algorithm 1.

Algorithm 1: DRL-AMIR for Flow Scheduling in ZTN

Input: Source node v_s, destination node v_d, learning rate α, reward discount factor γ, sampling size k, target network update frequency F_t, number of iterations I_n, expert discount factor D.
Output: Optimal path from v_s to v_d.
Initialize the network $Q_w(s, a)$ with random parameters w, where s is the state and a is the action.
Initialize the target network Q_{w^-} by copying the parameters $w^- \leftarrow w$.
Initialize the experience replay buffer R_b
Initialize the topology map M_{topo} using information collected from the SDN controller.
1. **For** episode = 1 to I_n **do:**
2. Generate the initial state s_t.
3. **While** True **do:**
4. Select an action a_t using the decaying ϵ-greedy policy.
5. **if** executing a_t results in an anomaly:
6. Assign a penalty r_t.
7. Store (s_t, a_t, r_t, s_{t+1}) as an experience into experience pool E_p.
8. Adjust the expert coefficient.
9. Use the expert mechanism to select an action a_t.
10. Execute a_t, obtain r_t and next state s_{t+1}.
11. Store (s_t, a_t, r_t, s_{t+1}) as an experience into E_p.
12. **if** the number of experience in E_p exceeds the sampling size k:
13. Sample k experiences from the E_p.
14. Compute the target value for experience i: $y_i = r_i + \gamma max Q_{w^-}(s_{i+1}, a)$.
15. Minimize the target loss.
16. **if** the update frequency F_t is met:
17. Update the target network.
18. **if** the destination node v_d is reached after executing action a_t:
19. Break the loop.
20. **if** the number of loop iterations exceeds the total number of nodes in the network:
21. Break the loop.
22. Update the current state s_t to s_{t+1}.
23. **End While**
24. **End For**

5 Experimental Setup and Performance Evaluation

To evaluate the effectiveness of the proposed DRL-AMIR routing method, we design a comprehensive set of comparative experiments. In this section, we first introduce the experimental setup, including the runtime environment, network topology, and comparative methods. Subsequently, we present the specific results and analyses of the proposed DRL-AMIR method in terms of hyperparameter setting and network performance.

5.1 Experimental Setup

The experimental environment is set up on a server equipped with a 16-core CPU, 64GB of RAM, and an NVIDIA 3090 GPU. On this server, we install the Ubuntu 16.04 operating system and utilize the Mininet network emulator along with the Ryu controller to construct the ZTN environment.

To test the effectiveness of the proposed DRL-AMIR method in different network environments, this paper employs three network topologies of varying scales. These three network topologies [31] consist of 10, 14, and 21 network nodes, respectively.

To evaluate the performance of DRL-AMIR in terms of network performance, this paper conducts comparative experiments with four existing methods, namely SPR, LBR, QoSR, and DRSIR [25]. The experimental methodology is defined as follows: (1) SPR constructs paths with minimal network hop counts for routing tasks; (2) LBR composes paths using links with maximum residual bandwidth; (3) QoSR minimize hop counts for delay-sensitive flows and maximize residual bandwidth for throughput-sensitive flows; (4) DRSIR implements a DQN-driven approach to intelligently identify optimal transmission paths.

5.2 Hyperparameter Setting

In the training of the proposed DRL-AMIR, hyperparameters such as the learning rate, sampling size, target network update frequency, and discount factor play a critical role in determining model performance. By tuning these parameters and observing the convergence speed of the final reward as well as the reward value after convergence, we identified the optimal parameter settings for our network environment. The experimental results of testing these hyperparameters are illustrated in Fig. 2.

As shown in Fig. 2a, testing values of 0.01, 0.001, and 0.0001 revealed that lr = 0.01 provides the fastest convergence (within 100 episodes) and highest stable reward (0.85). Lower rates showed significantly slower convergence, with lr = 0.001 requiring 200 episodes and lr = 0.0001 stabilizing only after 300 episodes with notable early fluctuations.

Fig. 2b demonstrates that with values of 0.5, 0.7, and 0.9 tested, $\gamma = 0.9$ showed superior performance, achieving rapid reward increase after 50 episodes and stabilizing near 0.85. This indicates that prioritizing long-term returns improves decision-making effectiveness. Lower values ($\gamma = 0.7$ and $\gamma = 0.5$) exhibited slower convergence and inferior overall performance.

As illustrated in Fig. 2c, comparing sizes of 32, 64, and 128, a batch size of 128 showed the highest stability during convergence. Though initial convergence was slightly slower, it produced a smooth learning curve with minimal fluctuations and consistently high reward values in later stages, indicating more comprehensive utilization of experience samples and more stable gradient updates.

Fig. 2d shows that testing values of 50, 100, and 500 revealed that update = 50 achieved optimal results in both convergence speed and reward optimization. This setting demonstrated quick convergence to high reward levels with smaller post-convergence fluctuations, effectively accelerating the learning process while maintaining training stability.

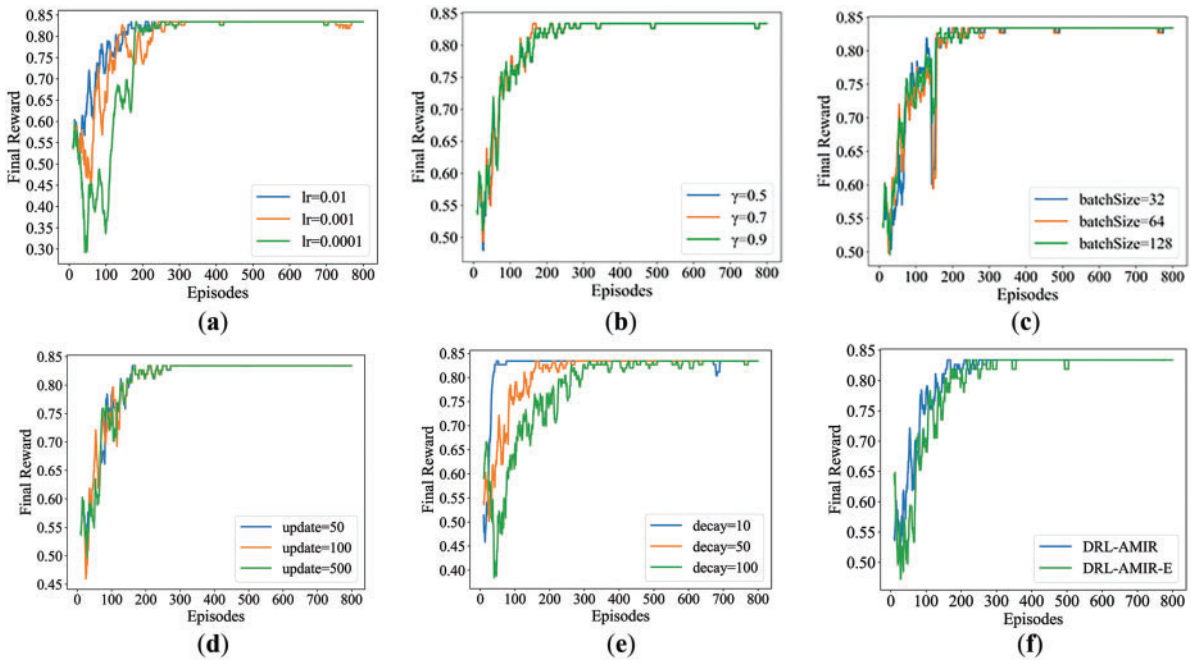

Figure 2: Hyperparameter test of the proposed DRL-AMIR: (**a**) learning rate; (**b**) discount factor γ; (**c**) batch size; (**d**) update; (**e**) decay-ε-greedy; (**f**) expert mechanism

As depicted in Fig. 2e, with convergence epochs set to 10, 50, and 100, decay = 10 showed rapid early convergence and stable high reward values. Larger decay values (50 and 100) resulted in slower convergence and greater fluctuations, with decay = 100 showing persistent instability even in later stages.

Fig. 2f compares training with and without the expert mechanism, demonstrating that DRL-AMIR with the mechanism achieves faster initial convergence, reaching stability after approximately 200 episodes with higher reward values. Without the mechanism, DRL-AMIR-E exhibited slower convergence and greater fluctuations, though eventually reaching similar reward levels. The expert mechanism thus effectively improves both convergence speed and model stability.

5.3 Network Performance

To evaluate the performance of the proposed DRL-AMIR method in terms of transmission delay, load balancing efficiency, and network bandwidth overhead, this section presents comparative analyses against four benchmark methods: SPR, LBR, QoSR, and DRSIR. The experimental results, as illustrated in Figs. 3–5, demonstrate the comparative performance metrics across these methodologies.

As can be seen from Fig. 3, DRL-AMIR demonstrates significant advantages in low latency across different topological scales and flow types. This indicates that the method can efficiently balance link load and latency requirements in path selection, exhibiting excellent latency optimization performance. DRL-AMIR adaptively adjusts its routing decisions based on the characteristics of different traffic types, showcasing strong generalization capabilities. Even in topologies with a larger number of nodes and more complex network structures, it maintains its low-latency advantage, further validating its applicability in complex scenarios. Additionally, compared to other routing methods, DRL-AMIR exhibits smaller fluctuations in latency, indicating more stable performance. This suggests that its routing decisions are more reliable in dynamic networks and less susceptible to changes in network topology or load fluctuations.

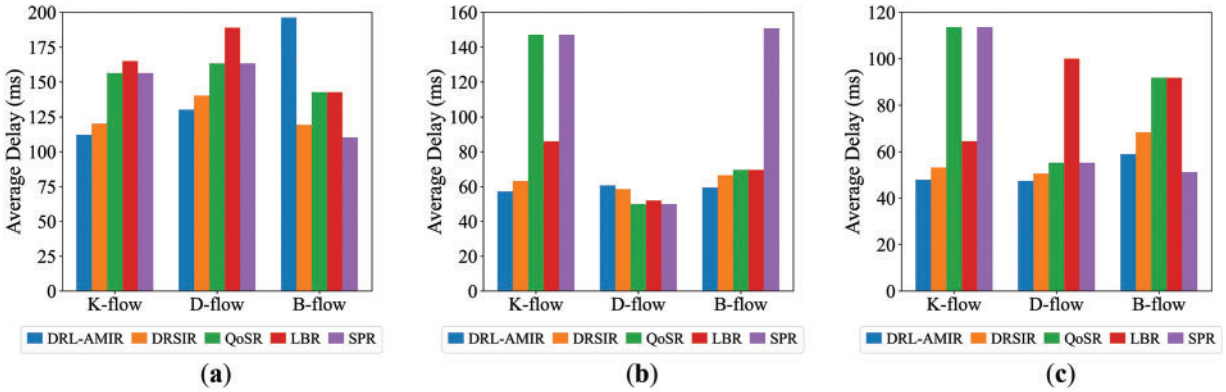

Figure 3: Average transmission delay of diverse flows in different network topology: (**a**) 10-nodes topology; (**b**) 14-nodes topology; (**c**) 21-nodes topology

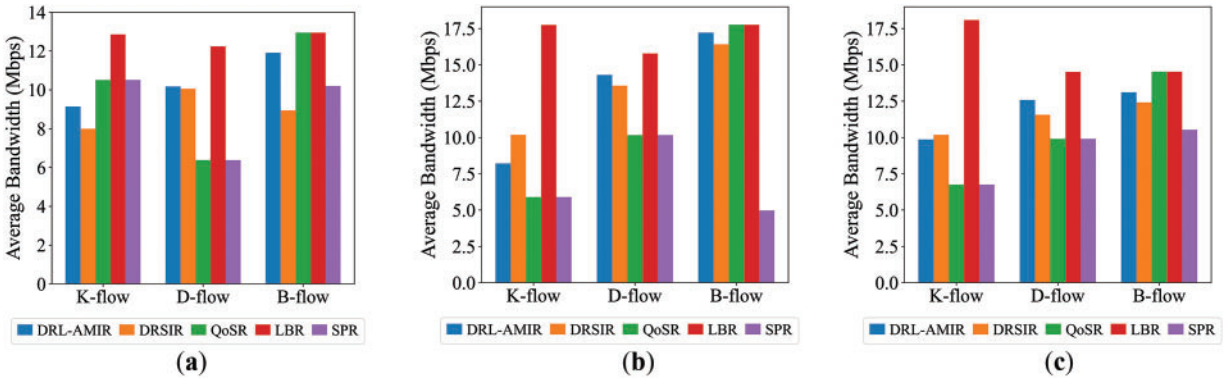

Figure 4: Average bottleneck bandwidth of diverse flows in different network topology: (**a**) 10-nodes topology; (**b**) 14-nodes topology; (**c**) 21-nodes topology

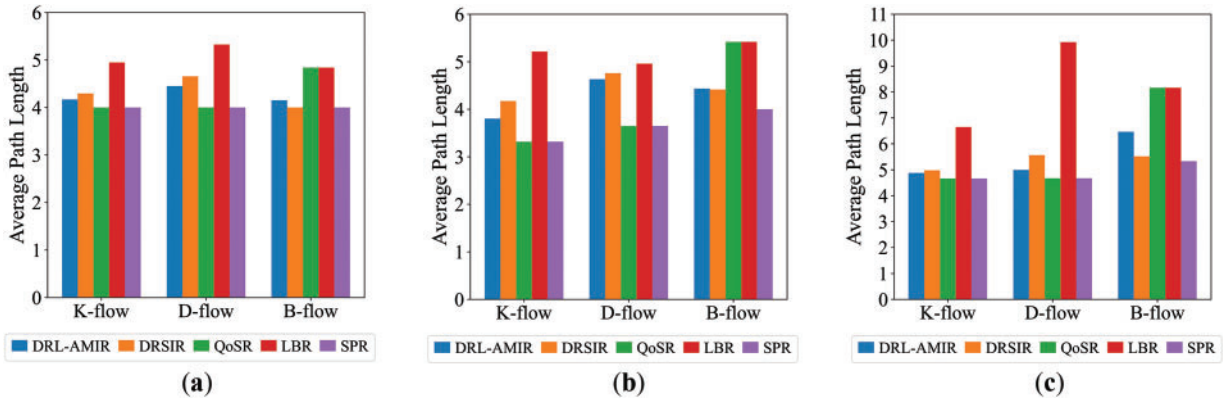

Figure 5: Average path length of diverse flows in different network topology: (**a**) 10-nodes topology; (**b**) 14-nodes topology; (**c**) 21-nodes topology.

From Fig. 4, it is evident that DRL-AMIR shows a clear advantage in bottleneck bandwidth for both D-flows and B-flows. This demonstrates that DRL-AMIR can effectively balance bottleneck bandwidth with other performance metrics, ensuring high resource utilization and meeting the transmission requirements of D-flows and B-flows. However, DRL-AMIR does not outperform all other methods in every scenario.

Specifically, compared to the LBR method, LBR achieves theoretically optimal results in terms of bottleneck bandwidth because it solely focuses on maximizing bottleneck bandwidth without considering other factors such as latency. In contrast, DRL-AMIR strikes a balance among multiple metrics, including bottleneck bandwidth, latency, and hop count, making it more practical for real-world applications.

From Fig. 5, it is apparent that DRL-AMIR performs well in reducing path hop count. Although SPR achieves the best performance among all methods by selecting the shortest path, which theoretically minimizes hop count, it only considers path length and fails to account for other critical network performance metrics such as bottleneck bandwidth and link load. This can lead to over-concentration of bandwidth resources in certain network environments, causing congestion or imbalanced resource allocation. In contrast, DRL-AMIR finds an optimal balance among multiple performance metrics, effectively reducing hop count while considering factors such as bandwidth, latency, and link load. This enables more efficient optimization of network resource utilization, particularly in dynamic scenarios with complex topologies and multiple flow types.

6 Conclusion

This article proposes DRL-AMIR, a novel flow scheduling method for software defined zero trust networks that addresses the key challenges of network flexibility and service quality. The main contributions of this work are threefold. Firstly, we propose a comprehensive flow scheduling optimization model tailored for the ZTN environment, which effectively considers the differentiated requirements of various network flow types in terms of latency, bandwidth, and path hop count in terms of network performance indicators. Secondly, we developed the DRL-AMIR method, which has a specially designed state space, action space, reward function, and adaptive expert mechanism, significantly improving the decision-making efficiency of flow scheduling in ZTN environments. Thirdly, through extensive experiments on various network topologies of different scales, we have demonstrated that DRL-AMIR consistently outperforms existing methods in key performance indicators, including SPR, LBR, QoSR, and DRSIR.

Future research will focus on integrating meta-learning techniques to enable rapid adaptation across dynamic network topologies. Additionally, we aim to develop hierarchical action space architectures with adaptive granularity control, enhancing DRL-AMIR's capability to handle abrupt environmental changes while maintaining computational efficiency.

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

Funding Statement: This work was supported in part by Scientific Research Fund of Zhejiang Provincial Education Department under Grant Y202351110, in part by Huzhou Science and Technology Plan Project under Grant 2024YZ23, in part by Research Fund of National Key Laboratory of Advanced Communication Networks under Grant SCX23641X004, and in part by Postgraduate Research and Innovation Project of Huzhou University under Grant 2024KYCX50.

Author Contributions: The authors confirm their contribution to the paper as follows: Conceptualization: Wenlong Ke, Zilong Li, Peiyu Chen; Methodology: Wenlong Ke, Zilong Li, Peiyu Chen; Software: Peiyu Chen, Jinglin Lv; Writing—original draft preparation: Wenlong Ke, Peiyu Chen; Writing—review and editing: Wenlong Ke, Zilong Li, Benfeng Chen, Qiang Wang, Ziyi Jia, Shigen Shen; Funding Acquisition: Wenlong Ke, Benfeng Chen. 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. Zhang J, Zheng J, Zhang Z, Chen T, Ya T, Zhang Q, et al. ATT&CK-based advanced persistent threat attacks risk propagation assessment model for zero trust networks. Comput Netw. 2024;245(2):110376. doi:10.1016/j.comnet.2024.110376.
2. Erel-Özçevik M. Token as a service for software-defined zero trust networking. J Netw Syst Manag. 2024;33(1):1–20. doi:10.1007/s10922-024-09894-w.
3. Xu X, Zhou X, Zhou X, Bilal M, Qi L, Xia X, et al. Distributed edge caching for zero trust-enabled connected and automated vehicles: a multi-agent reinforcement learning approach. IEEE Wirel Commun. 2024;31(2):36–41. doi:10.1109/mwc.001.2300414.
4. Hao M, Tan B, Wang S, Yu R, Liu RW, Yu L. Exploiting blockchain for dependable services in zero-trust vehicular networks. Front Comput Sci. 2023;18(2):182805. doi:10.1007/s11704-023-2495-0.
5. Rostami M, Goli-Bidgoli S. An overview of QoS-aware load balancing techniques in SDN-based IoT networks. J Cloud Comput. 2024;13(1):89. doi:10.1186/s13677-024-00651-7.
6. Hu M, Xiao M, Hu Y, Cai C, Deng T, Peng K. Software defined multicast using segment routing in LEO satellite networks. IEEE Trans Mob Comput. 2024;23(1):835–49. doi:10.1109/tmc.2022.3215976.
7. Xie H, Wang Y, Ding Y, Yang C, Liang H, Qin B. Industrial wireless internet zero trust model: zero trust meets dynamic federated learning with blockchain. IEEE Wirel Commun. 2024;31(2):22–9. doi:10.1109/mwc.001.2300368.
8. Hu M, Li J, Cai C, Deng T, Xu W, Dong Y. Software defined multicast for large-scale multi-Layer LEO satellite networks. IEEE Trans Netw Serv Manag. 2022;19(3):2119–30. doi:10.1109/tnsm.2022.3151552.
9. Dhanaraj RK, Singh A, Nayyar A. Matyas-Meyer Oseas based device profiling for anomaly detection via deep reinforcement learning (MMODPAD-DRL) in zero trust security network. Computing. 2024;106(6):1933–62. doi:10.1007/s00607-024-01269-y.
10. Dong T, Zhuang Z, Qi Q, Wang J, Sun H, Yu FR, et al. Intelligent joint network slicing and routing via GCN-powered multi-task deep reinforcement learning. IEEE Trans Cogn Commun Netw. 2022;8(2):1269–86. doi:10.1109/tccn.2021.3136221.
11. Bhavanasi SS, Pappone L, Esposito F. Dealing with changes: resilient routing via graph neural networks and multi-agent deep reinforcement learning. IEEE Trans Netw Serv Manag. 2023;20(3):2283–94. doi:10.1109/tnsm.2023.3287936.
12. Dhiman P, Saini N, Gulzar Y, Turaev S, Kaur A, Nisa KU, et al. A review and comparative analysis of relevant approaches of zero trust network model. Sensors. 2024;24(4):1328. doi:10.3390/s24041328.
13. Ye J, Cheng W, Liu X, Zhu W, Xa W, Shen S. SCIRD: revealing infection of malicious software in edge computing-enabled IoT networks. Comput Mater Contin. 2024;79(2):2743–69. doi:10.32604/cmc.2024.049985.
14. Shen Y, Shepherd C, Ahmed CM, Shen S, Yu S. SGD3QN: joint stochastic games and dueling double deep Q-networks for defending malware propagation in edge intelligence-enabled internet of things. IEEE Trans Inf Forensics Secur. 2024;19:6978–90. doi:10.1109/tifs.2024.3420233.
15. Okegbile SD, Cai J, Chen J, Yi C. Blockchain for secure data sharing in zero-trust human digital twin systems. In: Nguyen TA, editor. Blockchain and digital twin for smart hospitals. Amsterdam, The Netherlands: Elsevier; 2025. p. 337–61. doi:10.1016/b978-0-443-34226-4.00018-6.
16. Okegbile SD, Cai J, Chen J, Yi C. A reputation-enhanced shard-based byzantine fault-tolerant scheme for secure data sharing in zero trust human digital twin systems. IEEE Internet Things J. 2024;11(12):22726–41. doi:10.1109/JIOT.2024.3382829.
17. Singh A, Dhanaraj RK, Sharma AK. Personalized device authentication scheme using Q-learning-based decision-making with the aid of transfer fuzzy learning for IIoT devices in zero trust network (PDA-QLTFL). Comput Electr Eng. 2024;118(5):109435. doi:10.1016/j.compeleceng.2024.109435.
18. Pacini A, Scano D, Sgambelluri A, Valcarenghi L, Giorgetti A. Hybrid-hierarchical synchronization for resilient large-scale SDN architectures. IEEE Access. 2025;13(1):9032–46. doi:10.1109/access.2025.3527224.

19. Ali J, Song HH, Roh B-H. An SDN-based framework for E2E QoS guarantee in internet-of-things devices. IEEE Internet Things J. 2024;12(1):605–22. doi:10.1109/JIOT.2024.3465609.
20. Alenazi MJ, Ali J. An effective deep-Q learning scheme for QoS improvement in physical layer of software-defined networks. Phys Commun. 2024;66(2):102387. doi:10.1016/j.phycom.2024.102387.
21. Tan KY, Tan SC, Chuah TC. A multi-phase DRL-driven SDN migration framework addressing budget, legacy service compatibility, and dynamic traffic. IEEE Access. 2025;13(4):33202–19. doi:10.1109/access.2025.3543236.
22. Wang L, Lu L, Wang M, Li Z, Yang H, Zhu S, et al. SNS: smart node selection for scalable traffic engineering in segment routing networks. IEEE Trans Netw Serv Manag. 2025;22(1):92–106. doi:10.1109/tnsm.2024.3424928.
23. Shen S, Cai C, Shen Y, Wu X, Ke W, Yu S. MFGD3QN: enhancing edge intelligence defense against DDoS with mean-field games and dueling double deep Q-Network. IEEE Internet Things J. 2024;11(13):23931–45. doi:10.1109/jiot.2024.3387090.
24. Shen S, Cai C, Shen Y, Wu X, Ke W, Yu S. Joint mean-field game and multiagent asynchronous advantage actor-critic for edge intelligence-based IoT malware propagation defense. IEEE Trans Dependable Secur Comput. 2025;2025:1–15. doi:10.1109/TDSC.2025.3542104.
25. Casas-Velasco DM, Rendon OMC, da Fonseca NLS. DRSIR: a deep reinforcement learning approach for routing in software-defined networking. IEEE Trans Netw Serv Manag. 2022;19(4):4807–20. doi:10.1109/tnsm.2021.3132491.
26. He Q, Wang Y, Wang X, Xu W, Li F, Yang K, et al. Routing optimization with deep reinforcement learning in knowledge defined networking. IEEE Trans Mob Comput. 2024;23(2):1444–55. doi:10.1109/tmc.2023.3235446.
27. Shen S, Xie L, Zhang Y, Wu G, Zhang H, Yu S. Joint differential game and double deep Q-networks for suppressing malware spread in industrial internet of things. IEEE Trans Inf Forensics Secur. 2023;18:5302–15. doi:10.1109/tifs.2023.3307956.
28. Sun P, Guo Z, Li J, Xu Y, Lan J, Hu Y. Enabling scalable routing in software-defined networks with deep reinforcement learning on critical nodes. IEEE/ACM Trans Netw. 2022;30(2):629–40. doi:10.1109/tnet.2021.3126933.
29. Tao X, Monaco D, Sacco A, Silvestri S, Marchetto G. Delay-aware routing in software-defined networks via network tomography and reinforcement learning. IEEE Trans Netw Sci Eng. 2024;11(4):3383–97. doi:10.1109/tnse.2024.3371384.
30. Shen S, Hao X, Gao Z, Wu G, Shen Y, Zhang H, et al. SAC-PP: jointly optimizing privacy protection and computation offloading for mobile edge computing. IEEE Trans Netw Serv Manag. 2024;21(6):6190–203. doi:10.1109/tnsm.2024.3447753.
31. Ye M, Zhao C, Wen P, Wang Y, Wang X, Qiu H. DHRL-FNMR: an intelligent multicast routing approach based on deep hierarchical reinforcement learning in SDN. IEEE Trans Netw Serv Manag. 2024;21(5):5733–55. doi:10.1109/tnsm.2024.3402275.