IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 11 July 2023, accepted 31 July 2023, date of publication 4 August 2023, date of current version 10 August 2023.

*Digital Object Identifier 10.1109/ACCESS.2023.3302178*

RESEARCH ARTICLE

# An Intelligent SDWN Routing Algorithm Based on Network Situational Awareness and Deep Reinforcement Learning

**JINQIANG LI[1,2], MIAO YE[1,2,3], LINQIANG HUANG[4], XIAOFANG DENG[1,2], HONGBING QIU[1,2], YONG WANG[4], AND QIUXIANG JIANG[2]**

[1]School of Information and Communication, Guilin University of Electronic Technology, Guilin 541004, China
[2]Guangxi Key Laboratory of Wireless Broadband Communication and Signal Processing, Guilin University of Electronic Technology, Guilin 541004, China
[3]Ministry of Education Key Laboratory of Cognitive Radio and Information Processing, Guilin University of Electronic Technology, Guilin 541004, China
[4]School of Computer Science and Information Security, Guilin University of Electronic Technology, Guilin 541004, China

Corresponding author: Xiaofang Deng (xfdeng@guet.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 62161006 and Grant 62172095; in part by the Subsidization of Innovation Project of Guangxi Graduate Education under Grant YCSW2023310; in part by the Key Laboratory of Cognitive Radio and Information Processing, Ministry of Education, Guilin University of Electronic Technology, under Grant CRKL220103; and in part by the Guangxi Key Laboratory of Wireless Wideband Communication and Signal Processing under Grant GXKL06220110 and Grant GXKL06230102.

**ABSTRACT** To address the challenges of obtaining network state information, flexibly forwarding data, and improving the communication quality of service (QoS) in wireless network transmission environments in response to dynamic changes in network topology, this paper introduces an intelligent routing algorithm based on deep reinforcement learning (DRL) with network situational awareness under a software-defined wireless networking (SDWN) architecture. First, comprehensive network traffic information is collected under the SDWN architecture, and a graph convolutional network-gated recurrent unit (GCN-GRU) prediction mechanism is used to perceive future traffic trends. Second, a proximal policy optimization (PPO) DRL-based data forwarding mechanism is designed in the knowledge plane. The predicted network traffic matrix and topology information matrix are treated as the DRL environment, while next-hop adjacent nodes are treated as executable actions, and action selection policies are designed for different network conditions. To guide the learning and improvement of the DRL agent's routing strategy, reward functions of different forms are designed by utilizing network link information and different penalty mechanisms. Additionally, importance sampling steps and gradient clipping methods are employed during gradient updating to enhance the convergence speed and stability of the designed intelligent routing method. Experimental results show that this solution outperforms traditional routing methods in network throughput, delay, packet loss rate, and wireless node distance. Compared to value-function-based Dueling Deep Q-Network (DQN) routing, the convergence of the proposed method is significantly faster and more stable. Simultaneously, hardware storage consumption is reduced, and real-time routing decisions can be made using the current network state information. The source code can be accessed at https://github.com/GuetYe/DRL-PPONSA.

**INDEX TERMS** Deep reinforcement learning, gradient clipping, intelligent routing, importance sampling, network situational awareness, and software-defined wireless networking.

## I. INTRODUCTION

With the rapid development of wireless local area network (WLAN) technology and the exponential growth in the number of mobile terminals in operation, wireless access

The associate editor coordinating the review of this manuscript and approving it for publication was Paulo Mendes.

points (APs) are being provided in more locations to allow users to access the internet. This has resulted in explosive growth in wireless network traffic. In addition, the bursty traffic and rapid mobility of mobile wireless users can cause significant delay and packet loss issues in traditional wireless networks. This situation presents significant challenges for traditional wireless network architectures in terms of the physical infrastructure, protocol framework, and transmission performance, severely affecting the communication quality of service (QoS) for wireless users [1]. The efficient routing and transmission of data in wireless networks rely on the timely and flexible acquisition of network state information and the design of efficient routing algorithms. Therefore, researching ways to optimize the wireless network architecture and routing protocol design to improve transmission efficiency has important theoretical significance and practical value for maximizing the utilization of network resources in current wireless transmission scenarios.

In traditional wireless network frameworks, data forwarding and control management are tightly coupled, which limits network scalability. Consequently, network devices need to support multiple integrated network functions, leading to a cumbersome network architecture that hinders network management and maintenance [2]. Moreover, traditional distributed network architectures have deployment limitations. In the deployment of new network services in large-scale dynamic networks, complex and heterogeneous network structures can easily arise. To address these issues, software-defined wireless networking (SDWN) is considered a reliable solution that aims to improve the programmability and flexibility of wireless network architectures to facilitate more efficient network management and performance optimization. SDWN is an emerging wireless network framework that is an extension of software-defined networking (SDN). In this framework, the vertical structure of traditional wireless networks is broken up by separating the complex control functions of traditional wireless network devices and logically concentrating them on controllers to decouple data forwarding and control management. In this way, direct programmability and centralized control of network logic are realized along with the abstraction of the underlying wireless infrastructure, enabling network management through software. An SDWN controller schedules network resources from a global perspective and at a fine-grained level, using the southbound interface of the OpenFlow protocol and the data plane for programming control. Thus, the controller achieves the functions of issuing flow tables and collecting network status information. Meanwhile, the northbound interface of the controller provides an open interface with the application plane [3]. Based on the advantages outlined above, data forwarding and control management can be decoupled in SDWN, providing the possibility for routing algorithms to reschedule and efficiently utilize network resources.

Efficient routing protocol design is crucial in wireless networks, and the optimization of routing algorithms is a key area of research in traffic engineering [4]. A routing algorithm aims to find the most efficient routing strategy from the source node to the destination node in the network topology. The algorithm's performance determines the QoS in the wireless network. Consequently, designing a routing algorithm that is both efficient and stable is crucial for enhancing overall performance and resource utilization in an SDWN environment.

In recent years, researchers have enhanced network performance by optimizing routing algorithms. Traditional routing algorithms, such as the distance vector routing protocols (DVRPs) [5] and open shortest path first (OSPF) algorithms [6], are primarily based on the shortest-path approach. However, these algorithms often use only limited link information for optimization and cannot fully utilize the global network information to enhance network performance [3]. Moreover, traditional routing algorithms have shortcomings such as a slow convergence speed, a long response time, and difficulty adapting to dynamic networks, which make them unsuitable for use in an SDWN architecture. Some scholars have modeled the routing optimization problem as either a linear programming (LP) problem or a nonlinear programming (NLP) problem [7] and proposed the use of classical routing optimization algorithms to find the optimal routing strategy. However, as the network scale increases, it becomes difficult for such an algorithm to dynamically adjust the routing and forwarding strategy based on the changing network topology and link information. Some researchers have proposed improving the effectiveness and stability of routing algorithms by performing predictive processing on the available network traffic information before routing optimization [8]. By doing so, accurate traffic information can be obtained promptly to help the routing algorithm make precise path decisions. With the development of artificial intelligence technology, many researchers have begun to apply related methods to routing optimization problems to improve the performance of large-scale and complex dynamic networks and simultaneously optimize multiple network links. In particular, deep reinforcement learning (DRL) [9] is useful for solving goal-oriented learning and decision-making problems and thus can help routing algorithms flexibly adapt to complex, dynamic changes in wireless networks for the optimize the allocation of wireless network resources. For the wireless network routing problem discussed in this article, DRL utilizes deep neural networks that can handle high-dimensional and complex state and action spaces, better adapt to a dynamic network environment, and adaptively adjust the network routing strategy to maximize the overall network efficiency. In previous work, DRL has already achieved some significant results in routing optimization [9], [10], [11], [12].

This article proposes an intelligent SDWN routing algorithm, DRL-PPONSA, based on proximal policy optimization (PPO) DRL and network situational awareness (NSA) [10]. First, this algorithm comprehensively considers the traffic information on the links in the wireless network.

To this end, the data plane and control plane of the SDWN architecture and a graph convolutional network-gated recurrent unit (GCN-GRU) [11] prediction mechanism are designed as three key components of the NSA system. The control plane is responsible for data perception and acquisition, while the GCN-GRU mechanism is applied for traffic prediction, thereby achieving awareness of the global network state. Next, traffic matrices (TMs) representing the predicted remaining bandwidth, link delay, packet loss rate, packet error rate, and wireless node distance as well as the network topology are taken as the description of the environment for DRL, while next-hop adjacent nodes are viewed as possible actions. By utilizing the link information in the network environment and employing various reward and punishment mechanisms, a reward function is designed to guide a DRL agent to learn a strategy seeking the highest reward, ultimately achieving optimal intelligent routing decisions.

The contributions of this paper are summarized as follows:

1) This paper comprehensively considers multiple types of status information, such as network delay, network jitter, remaining bandwidth, packet loss rate, packet error rate, and distance between APs, and uses them as optimization objectives. This approach can more comprehensively satisfy QoS requirements, laying a foundation for efficient and flexible data transmission and effectively reducing network congestion.
2) This paper presents an NSA mechanism using a GCN-GRU prediction model based on an SDWN architecture. This mechanism comprehensively considers the spatiotemporal characteristics of wireless network traffic, using current and historical traffic status data to predict the future traffic situation. This approach can fully utilize the hidden status information in a wireless network to improve the reliability and stability of an intelligent routing algorithm.
3) By designing the action space for DRL to consist of the next-hop adjacent nodes, the interactivity between the intelligent agent and the network environment is enhanced. This enables the routing algorithm to more intelligently adapt to dynamic and complex changes in the network. Additionally, different reward functions are designed based on link information and reward mechanisms to mitigate the issue of the intelligent agent becoming stuck in local optima. In this way, the robustness and effectiveness of the intelligent routing algorithm are enhanced.
4) An online DRL method based on PPO is employed to address the issues of low effectiveness of data samples due to mismatch with the current agent policy and gradient explosion during the gradient update process. To overcome these issues, importance sampling (IS) and gradient clipping techniques are utilized instead of prioritized experience replay (PER) and the Kullback-Leibler (KL) divergence. This approach improves the convergence speed and stability of the designed algorithm.

The remainder of this article is organized as follows: Section II introduces mainstream prediction models and routing methods along with their associated shortcomings. Section III introduces the proposed intelligent routing architecture based on NSA under the SDWN architecture. Section IV introduces the environment design for the proposed DRL-PPONSA algorithm and the details of algorithm implementation for intelligent routing. Section V reports experiments conducted to verify the stability and reliability of the algorithm proposed in this paper. Section VI is the conclusion, which summarizes the challenges faced and proposes further research directions.

## II. RELATED WORK

This section introduces commonly used routing optimization methods and traffic prediction methods used to improve the reliability and stability of routing algorithms. The mainstream routing optimization methods are divided into classical optimization algorithms and intelligent optimization methods, while the traffic prediction methods are also divided into two main categories: linear prediction and nonlinear prediction.

### A. ROUTING OPTIMIZATION METHODS

Routing optimization is a key technology for improving network performance, providing important guarantees for network scalability, stability, and security.

#### 1) CLASSIC ROUTING OPTIMIZATION METHODS

This section introduces classical route optimization algorithms based on heuristic algorithms. Common algorithms of this kind include simulated annealing (SA), genetic algorithms (GAs), ant colony optimization (ACO), and artificial neural networks (ANNs). As summarized in Table 1, several research papers [14], [15], [16], [17], [18], [19] have shown that classical routing optimization methods can improve routing performance in global optimization problems. However, they may face slow convergence when dealing with complex problems and large optimization spaces. Additionally, in paper [19], a long short-term memory (LSTM)-based traffic prediction method is used. However, compared to the GRU structure, the LSTM network structure is more complex and requires more parameters to be set, making it less convenient in terms of parameter tuning. Consequently, a significant amount of time is required to train such a prediction model. Moreover, LSTM prediction methods are not suitable for wireless network traffic with both temporal and spatial features. The scenario addressed in paper [14] involves wireless sensor networks, and the SDN architecture is not utilized, which limits the acquisition of global network information. While the methods presented in other papers [15], [16], [17], [18], [19] do employ the SDN architecture to improve routing, they cannot construct efficient routes based on dynamically changing network information, making them less applicable to complex dynamic network environments.

Although classical optimization algorithms can solve shortest-path optimization problems to some extent, these

algorithms typically provide only suboptimal solutions and often optimize only a single objective. Subtle changes in the network environment can cause significant fluctuations and errors in these algorithms, leading to potential scalability issues and impacting network performance [8]. Therefore, classical routing optimization algorithms are not suitable for large-scale, dynamic, and complex SDWN environments.

#### 2) INTELLIGENT ROUTING OPTIMIZATION METHODS

With the continuous development of science and technology, traditional classical routing algorithms are increasingly unable to meet complex and dynamic network demands. In contrast, intelligent optimization algorithms can better adapt to complex and high-dimensional dynamic network environments. Therefore, many researchers have studied the application of intelligent algorithms to routing optimization problems and achieved good results. As shown in Table 1, a supervised machine learning approach is used in paper [4] to improve routing in SDWN. However, this method requires a large amount of labeled data and is highly dependent on the data labels. Furthermore, the model learned during the training process may not generalize well to unseen data. Paper [20] employs the particle swarm optimization algorithm to solve the routing problem in SDWN. Due to the algorithm's randomness and the local search capability of the swarm, this algorithm is prone to becoming stuck in local optima and struggles to escape and find the global optimum. On the other hand, papers [21], [22] utilize reinforcement learning (RL) to optimize route selection. In contrast to supervised machine learning, RL involves interacting with the environment and improving routing strategies by optimizing their reward values, thereby eliminating the need for a large amount of labeled data. However, when dealing with high-dimensional action and state spaces, RL-based routing methods are not suitable. Therefore, some papers [3], [8], [23], [24], [25], [26], [27], [28] suggest using DRL to overcome the limitations of traditional RL methods. They utilize neural networks to estimate and extract features from high-dimensional actions and states. However, most of these papers primarily focus on optimizing network latency and packet loss while neglecting other factors such as bandwidth, error rates, and distance. Moreover, only a few papers consider prediction mechanisms, and most of them rely on prediction methods such as GRU networks. Among papers [23], [24], [25], [26], [27], [28], the majority adopt the k-paths action selection strategy, which often requires the use of traditional routing algorithms such as Dijkstra's algorithm to compute k paths that meet the given conditions. This leads to higher computational complexity and resource consumption compared to the next-hop strategy.

In the abovementioned literature, classical machine learning algorithms were used to optimize network routing and improve network performance. Although these approaches can optimize multiple target parameters simultaneously, doing so requires a large amount of labeled data for training, and obtaining corresponding label information from dynamic and complex network topologies is extremely difficult and highly dependent on the accuracy of the data. If the obtained dataset is inaccurate, the system will not be able to learn a good network model to predict the optimal routing strategy. Therefore, many researchers have begun to focus on reinforcement learning as an optimization method for intelligent routing. Compared with traditional routing algorithms and classical optimization algorithms, reinforcement learning methods can greatly improve network performance and have significant advantages in adjusting routing strategies in dynamically changing networks in a timely manner. Therefore, reinforcement learning is also a feasible technology for solving routing optimization problems in large-scale and complex dynamic wireless networks. However, most of the related literature to date has focused on intelligent routing problems in wired SDN environments, and there has been little in-depth discussion of wireless network routing.

### B. TRAFFIC PREDICTION METHODS

Network traffic information is an important feedback indicator of network load. Accurately and reliably predicting network traffic information can help avoid delays caused by network congestion and assist routing algorithms in achieving fault-tolerant processing, thus improving their stability and reliability. Typically, the algorithms used for network state prediction can be divided into linear prediction methods and nonlinear prediction methods.

#### 1) LINEAR PREDICTION METHODS

The most commonly used linear prediction models include autoregressive moving average (ARMA) models, autoregressive integrated moving average (ARIMA) models, Markov models, and the Holt-Winters algorithm. Tian et al. [29] proposed modeling the frequency component of network traffic with an ARMA model to predict network traffic in an SDN environment. Alghamdi et al. [30] proposed using an ARIMA model to predict traffic flow in the state of California in the United States. Tran et al. [31] proposed an improved algorithm based on Holt-Winters exponential smoothing to predict abnormal data in cellular traffic environments. However, traditional linear models have simple structures and have difficulty accurately perceiving fast-changing network traffic features, resulting in poor prediction results as well as weak model adaptability and generalization ability. The traffic information in wireless networks includes not only temporal features but also spatial features [21]. Therefore, using linear models to predict wireless traffic has certain limitations.

#### 2) NONLINEAR PREDICTION METHODS

For complex and diverse types of network data, accurate prediction can be achieved through nonlinear modeling; for example, nonlinear models such as neural networks and fuzzy logic models can be used to predict network traffic. Casado-Vara et al. [32] proposed using an LSTM recurrent

**TABLE 1. Related literature on routing approaches.**

| Paper | Description | Prediction mechanism | Learning approach | Performance metrics | Action | Data transmission scenario |
|---|---|---|---|---|---|---|
| [14] | A routing optimization strategy for wireless sensor networks based on improved genetic algorithm | None | Genetic algorithms (GAs) | Energy consumption, delay | Next-hop | Wireless sensor networks(WSN) |
| [15] | Discover the optimal IoT packets routing path of software-defined network via artificial bee colony algorithm | None | Bee colony optimization (BCO) | Distance | Next-hop | SDN |
| [16] | Optimization strategy of SDN control deployment based on simulated annealing-genetic hybrid algorithm | None | Simulated annealing (SA) and Genetic algorithm (GA) | Delay | Link weight values | SDN |
| [17] | A heuristic traffic engineering algorithm for use in SDN networks with multipath forwarding and inter-path traffic switching | None | Not reported | Paths load, total packet drop and flow properties | K-paths selection | SDN |
| [18] | FH-ACO: Fuzzy heuristic-based ant colony optimization for joint virtual network function placement and routing | None | Fuzzy heuristic-based ant colony optimization (FH-ACO) | Delay, reliability | Not reported | Virtual wired network |
| [19] | SDPredictNet-a topology based SDN neural routing framework with traffic prediction analysis | LSTM | Artificial neural networks (ANNs) | Bandwidth, delay | Link weight values | SDN |
| [4] | MLaR: machine-learning-assisted centralized link-state routing in software-defined-based wireless networks | None | ML (Machine learning) | Latency, bandwidth, SNR, distance | K-paths selection | SDWN |
| [20] | An energy-efficient routing algorithm for software-defined wireless sensor networks | None | (PSO) Particle swarm optimization | Energy of the network nodes | Not reported | SDWN |
| [21] | An intelligent routing algorithm based on machine learning in software-defined wireless networking | None | RL (Q-learning) | Delay, transmission quality | Link weight values | SDWN |
| [22] | Adaptive Routing in Wireless Mesh Networks Using Hybrid Reinforcement Learning Algorithm | None | RL (Q-learning) | Throughput, packet delivery ratio, and delay | Next-hop | Wireless mesh networks |
| [3] | DRL-M4MR: An intelligent multicast routing approach based on DQN deep reinforcement learning in SDN | None | DRL(DQN) | Delay, bandwidth, loss | Next adjacent edge | SDN |
| [8] | Intelligent routing method based on Dueling DQN reinforcement learning and network traffic state prediction in SDN | GRU | DRL (DQN) | Bandwidth, delay, loss | K-paths selection | SDN |
| [23] | An approach to combine the power of deep reinforcement learning with a graph neural network for routing optimization | None | DRL (DQN) and GNN | Path delay and link usage | Link weight values | Wired network |
| [24] | A DRL-based solution for intelligent routing in SDN based on path-state metrics | None | DRL (DQN) | Link stretch, delay, loss and throughput | K-paths selection | SDN |
| [25] | DRL-R: Deep reinforcement learning approach for intelligent routing in software-defined data-center networks | None | DRL (DDPG) | Delay | K-paths selection | SDN |
| [26] | A Deep Reinforcement Learning Approach for Deploying SDN Switches in ISP Networks from the Perspective of Traffic Engineering | None | DRL (DDPG) | Throughput | K-paths selection | SDN |
| [27] | Modeling Data Center Networks with Message Passing Neural Network and Multi-task Learning | None | DRL (DDPG) | Delay, network jitter, Drop ratio | K-paths selection | SDN |
| [28] | A scalable deep reinforcement learning approach for traffic engineering based on link control | GRU | DRL (DDPG) | Delay | K-paths selection | Traffic network |
| DRL-PPONSA | An SDWN intelligent routing solution based on PPO deep reinforcement learning and network situation awareness | GCN-GRU | DRL (PPO) | Bandwidth, delay, loss, packet error rate, AP distance | Next-hop | SDWN |

neural network trained through distributed asynchronous training to predict web traffic time series, fully exploring the hidden temporal features of the network traffic and improving the accuracy of prediction. Hu et al. [33] proposed using an improved variant of an LSTM recurrent neural network model in which the forget gate and input gate are combined into an update gate to predict data information in network traffic. Bi et al. [34] proposed using a Savitzky-Golay (SG) filter to preprocess the noise information in the original traffic data, using a temporal convolutional network (TCN) to extract the short-term features of the sequence, and using an LSTM network to capture the long-term dependencies in the data, effectively capturing the nonlinear features of the network sequence and improving the prediction accuracy. Huang et al. [8] proposed using a GRU model instead of an LSTM model to predict traffic matrix information in the SDN context, extracting hidden traffic information from the network to construct a corresponding predicted traffic matrix, and using the predicted traffic matrix to train an intelligent agent, thereby improving the reliability of the routing algorithm.

Although related studies have addressed the issues of low linear prediction accuracy and weak generalization ability and have achieved good prediction results, they have only considered the temporal features of network traffic and have not taken spatial features into account. For wireless network traffic with spatiotemporal characteristics, the prediction effect is not ideal. Therefore, Yuan et al. [35] proposed a method combining a 3D convolutional neural network (3D-CNN) and an LSTM network to predict traffic information in wireless networks. Pan et al. [36] fully considered the complex temporal and spatial dependencies among network traffic and proposed a prediction model combining the GCN and GRU techniques to predict network traffic. The experimental results showed that this model has long-term prediction capabilities and is applicable for traffic data with spatiotemporal characteristics, and the prediction effect meets expectations. Therefore, in this paper, a network traffic prediction scheme is designed that also combines the GCN and GRU approaches for the accurate prediction of wireless traffic with spatiotemporal features. The integration of this GCN-GRU prediction algorithm into the intelligent routing algorithm proposed in this paper is an important means of improving its reliability and effectiveness.

## III. THE DESIGN OF THE PROPOSED INTELLIGENT ROUTING ARCHITECTURE BASED ON SDWN FOR NSA

In this section, we introduce the proposed intelligent routing architecture designed in this paper based on SDWN for NSA.

The overall architecture of the intelligent routing system is composed of an infrastructure data plane, a logical control plane, a knowledge plane, and an application plane, as shown in Fig. 1. The structure of each plane is presented in detail below.

### A. INFRASTRUCTURE DATA PLANE

This plane primarily consists of wireless network devices such as APs and stations (STAs). This plane is responsible for packet lookup and forwarding as well as packet parsing and flow table matching. The devices respond to requests sent by the logical control plane via the southbound interface of the OpenFlow protocol.

### B. LOGICAL CONTROL PLANE

SDWN is a method of utilizing SDN technology to achieve centralized control in wireless networks. Accordingly, rules defined by specialized programs, usually referred to as controllers, determine the network behavior. In SDWN, the wireless control plane and data plane are decoupled and separated, thereby simplifying the wireless access devices and forwarding devices and allowing the network to run in accordance with the rules of logical centralized control plane scheduling. First, the SDWN controller in the control plane periodically collects information from the data plane, such as the network topology, link bandwidths, link round-trip delays, link packet loss rates, and distances between wireless APs, through its southbound interface.

The main functions of the controller include link discovery, topology management, policy formulation, and flow table distribution. For link discovery and topology management, the controller utilizes an uplink channel to uniformly monitor and aggregate the information reported by the underlying switch devices. Policy formulation and flow table distribution are achieved through the use of downstream channels to implement unified control over the network devices.

The purpose of link discovery in SDWN is to obtain global network information to serves as the foundation for network address learning, VLAN management, and routing/forwarding [37]. Unlike traditional network link discovery methods, link discovery in SDWN networks is accomplished uniformly through the link layer discovery protocol (LLDP) by the controller. The objective of topology management is to collect and monitor information from the SDWN switches in the network in order to provide feedback on the working status of the switches and the connection status of the links [38]. The controller regularly sends LLDP packets through packet-out messages to the connected SDWN APs and senses the information of these wireless APs based on the feedback obtained from packet-in messages, thereby detecting the working status of the wireless switches and updating its representation of the network topology.

Unlike in most other literature that considers only a single type of network state information, the network state information addressed in this article includes the remaining link

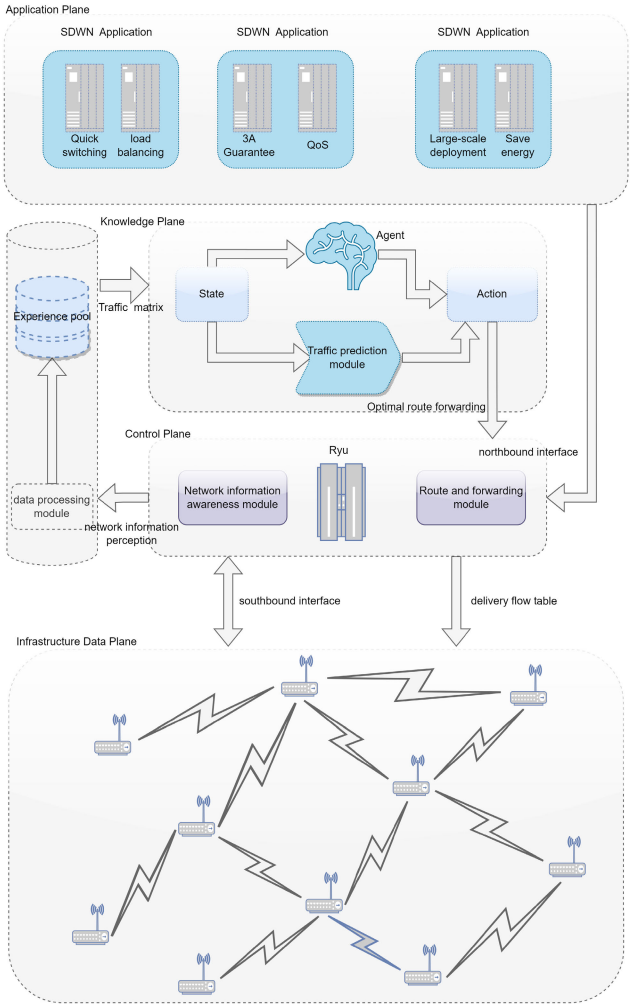



**FIGURE 1.** SDWN intelligent routing structure.

bandwidth $bw_{free(ij)}$, the used link bandwidth $bw_{use(ij)}$, the link delay $delay_{ij}$, the packet error rate $pkt_{err(ij)}$ on a link, the number of dropped packets $pkt_{drop(ij)}$ on a link, the distance $distance_{ij}$ between wireless APs, and the packet loss rate $loss_{ij}$ on a link. The controller operates with a cycle time of $t$ seconds and sends port information query requests to the data plane. The data plane responds with a reply that contains the statistics for each port, including the numbers of bytes transmitted $tx_b$ and received $rx_b$, the number of packets transmitted $tx_p$ and received $rx_p$, as well as the effective time $t_d$. The difference between two consecutive sets of statistics represents the amount of bandwidth used during that period. For a link $e_{(i,j)}$, the maximum used bandwidth $bw_{use(ij)}$ is determined by taking the maximum value of the used bandwidth between the two ports ($i$ and $j$) on the two switches connected by the link. The remaining bandwidth $bw_{free(ij)}$ on the link is equal to the difference between the maximum capacity of the link, $bw_{capmax}$, and the used bandwidth, $bw_{use(ij)}$, as shown in Equation (1). Equation (2) is used to calculate the maximum used bandwidth of the link, where $(i, j) \in (1, 2, 3, \cdots, n)$, $i \neq j$, the $n$ represent the number of

AP switches.

$$bw_{free(ij)} = \left| bw_{capmax} - bw_{use(ij)} \right| \tag{1}$$

$$bw_{use(ij)} = \left| \frac{\left| \left( tx_{bj} + rx_{bj} \right) - \left( tx_{bi} + rx_{bi} \right) \right|}{t_{dj} - t_{di}} \right| \tag{2}$$

The round-trip delay calculated for a link is detected by the controller's built-in Switches module, and the timestamp of LLDP [38] data transmission is obtained. The controller sends an echo request message with a timestamp to the AP switches, then parses the echo reply message returned by each switch and subtracts the transmission time parsed from the data from the current time to obtain the echo round-trip delays $T_{echo\_api}$ and $T_{echo\_apj}$ between the controller and the wireless switches. The delay between the wireless switches is then calculated from the LLDP message receiving time minus the message sending time for each switch, denoted by $T_{lldp\_api}$ and $T_{lldp\_apj}$. Accordingly, the $delay_{ij}$ calculation for link $e_{(i,j)}$ is as show in (3).

$$delay_{ij} = \frac{\left( T_{lldp\_api} + T_{lldp\_apj} - T_{echo\_api} - T_{echo\_api} \right)}{2}. \tag{3}$$

Equation (4) represents the maximum packet loss rate calculated for link $e_{(i,j)}$ from switch port $i$ to $j$ and from switch port $j$ to $i$. Equation (5) represents the packet error rate of link $e_{(i,j)}$ in both directions, from switch port $i$ to $j$ and from switch port $j$ to $i$. In Equation (6), the number of dropped packets for link $e_{(i,j)}$ represents the number of packets lost in both directions, from switch port $i$ to $j$ and from switch port $j$ to $i$.

$$loss_{ij} = \max\left( 1 - \frac{rx_{pj}}{tx_{pi}}, 1 - \frac{rx_{pi}}{tx_{pj}} \right) \cdot 100\% \tag{4}$$

$$pkt_{err(ij)} = \left| \frac{rx_{bi} - tx_{bj}}{rx_{bi}} \right| \cdot 100\% \tag{5}$$

$$pkt_{drop(ij)} = \left| rx_{pi} - tx_{pj} \right| \tag{6}$$

Here, $tx_{bi}$ represents the number of bytes sent from port $i$, $rx_{bi}$ represents the number of bytes received at port $i$. $tx_{pi}$ represents the number of packets sent from port $i$, $rx_{pi}$ represents the number of packets received at port $i$, $(i, j) \in (1, 2, 3, \cdots, n)$, $i \neq j$, and $n$ represent the number of AP switches.

In actual physical scenarios, the distance traveled corresponds to the energy consumed for wireless transmission; therefore, the algorithm in this article also considers the distance between APs. The Equation (7) gives the distance $distance_{ij}$ between two APs, which represents the distance in physical space between the wireless AP switches and is calculated using three-dimensional coordinates.

$$distance_{ij} = \sqrt{\left( x_i - x_j \right)^2 + \left( y_i - y_j \right)^2 + \left( z_i - z_j \right)^2} \tag{7}$$

Here, $(x_i, y_i, z_i)$ represents the spatial three-dimensional coordinates between of the i-th switch.

The controller formulates forwarding strategies and generates corresponding flow table entries based on the transmission requirements at the different levels. A designed DRL mechanism in the knowledge plane learns the optimal routing strategy based on the current network state information and issues flow tables to generate optimized SDWN forwarding routes.

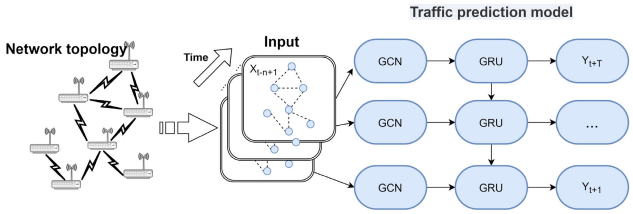



**FIGURE 2.** Traffic prediction module.

### C. KNOWLEDGE PLANE

The knowledge plane designed in this article applies a combination of RL based on policy gradients and traffic prediction. Loading the knowledge plane into the SDWN framework endows the controller with greater intelligence when making policy decisions. Model training in the knowledge plane requires data obtained from the information pool in advance and network link information converted into a traffic matrix, which is predicted by a traffic prediction module. The traffic prediction module consists of two parts: a GCN and a GRU network. As shown in Fig. 2, the topology information of the SDWN network is first obtained, and the Ryu controller obtains the link information and converts it into graph data with weights, which are treated as time series data at $t$ time points. Then, these time series data at $t$ historical time points are used as input for prediction. First, the spatial features of the wireless network topology are obtained through the GCN. Second, the time series data with spatial features are input into the GRU network for the extraction of temporal features. Finally, a fully connected layer is used to filter and output the predicted results. The obtained predicted traffic data are synchronized into a RL network for training, and an intelligent agent optimizes the strategy to be executed based on the currently obtained network state information, with the maximum reward value as the target, dynamically adjusting the optimal routing path until the model converges and stabilizes. Finally, the trained model is deployed to the SDWN controller to obtain the optimal routing strategy. Data flow table forwarding is completed by responding to the data plane through the southbound interface.

### D. APPLICATION PLANE

The application plane is a system that manages all business-related applications and has a programmable API. Controllers interact with the application plane through their northbound interfaces to enable the development and deployment of various network applications, such as load balancing, fast switching, and interference management.

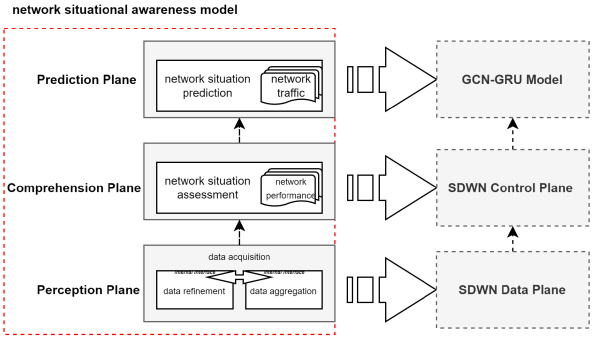



**FIGURE 3.** Architecture of network situation awareness system.

### E. DESIGN OF THE PROPOSED NSA SYSTEM ARCHITECTURE BASED ON SDWN

To enable the proposed intelligent routing algorithm to better optimize the network performance with enhanced stability and effectiveness, this article presents the design of an NSA architecture based on SDWN. This NSA architecture is used to monitor and obtain information on the traffic and network topology in the wireless network, predict future trends in network traffic, analyze network conditions from a global perspective, and provide accurate data support for routing. Meanwhile, DRL methods are used to search for the optimal wireless routing/forwarding paths to improve network bandwidth utilization and reduce the network delay, thereby ensuring the communication quality of the entire network.

Situational awareness refers to the ability to perceive environmental factors and events in time or space as well as predict their future states [39] by processing existing information and seeking an optimal solution on a specific timeline. The application of NSA in intelligent routing mainly involves perceiving and analyzing real-time data from the wireless network to optimize and schedule the network topology and wireless network traffic, thereby improving the network throughput and response speed and providing important guarantees of network security. As shown in Fig. 3, the NSA system architecture proposed in this article consists of three layers: a perception layer, a comprehension layer, and a prediction layer. Among them, the perception layer mainly collects and processes data information, corresponding to the data plane in SDWN. The comprehension layer corresponds to the control plane in the SDWN framework; in this layer, the Ryu controller comprehends and projects data information from the perception layer and evaluates the current situation of the network environment. As the highest level of situational awareness, the prediction layer is responsible for perceiving and understanding various elements of the network environment and predicting future traffic trends; this layer corresponds to the traffic prediction model in this article.

## IV. INTELLIGENT SDWN ROUTING ALGORITHM BASED ON POLICY OPTIMIZATION, DRL-PPONSA

In this section, we first introduce the overall architecture of the proposed algorithm, then describe the design process of the network traffic prediction algorithm and the DRL-PPO algorithm with NSA, and finally describe the DRL-PPONSA algorithm framework in detail.

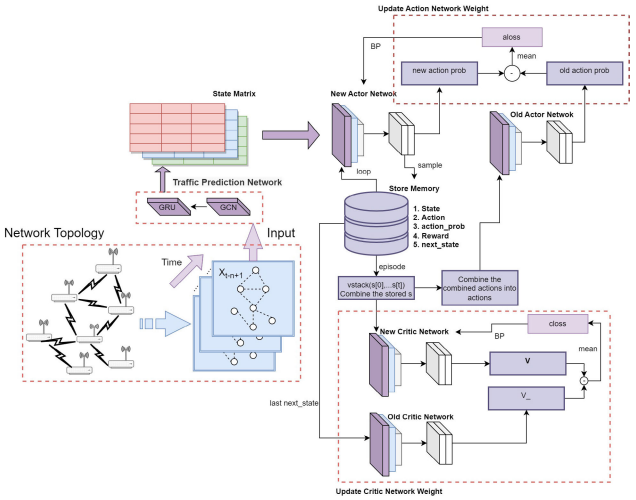



**FIGURE 4.** DRL-PPONSA algorithm framework.

### A. OVERALL ARCHITECTURE OF THE DRL-PPONSA ALGORITHM

In the infrastructure data plane, the wireless network topology is abstracted as an undirected graph with relevant link parameter weights, represented by $G = (G, E, W)$. Where $V$ is the set of wireless APs, $E$ is the set of links between nodes, $W$ is the set of link parameter information, and each side $e_{ij}$ represents a link, $e_{ij} \in E$. Accordingly, the intelligent routing algorithm in this article is designed based on an undirected graph. A diagram of the overall algorithm is shown in Fig. 4. First, the link information of the network is obtained based on the SDWN topology, including parameters such as bandwidth, delay, and packet loss rate, and is transformed into a multidimensional matrix. Then, the predicted traffic matrix is obtained through the traffic prediction model, and finally, the predicted traffic matrix is provided as input to the DRL algorithm for training. The DRL algorithm is composed of two main networks: an actor network and a critic network. The former optimizes a policy to maximize the expected returns, while the latter evaluates the value of a given policy in its current state. These two networks collaborate to achieve efficient policy optimization. This approach based on an AC network architecture can ensure that the intelligent agent always makes decisions in the direction of accumulating the maximum reward while dynamically optimizing parameters such as network delay, jitter, bandwidth, and throughput rate to achieve real-time control of the current network, thereby effectively reducing the network load [40].

### B. DESIGN OF THE NETWORK TRAFFIC PREDICTION ALGORITHM

This subsection mainly introduces the design process of the GCN-GRN prediction algorithm for achieving NSA.

**Algorithm 1** GCN-GRU Network Traffic State Prediction Algorithm

**Input:** traffic matrix data:$X_{1,2,3,\ldots,t}$, graph adjacency matrix:$A$, learning rate:$\alpha$, training episodes:$\mathcal{M}$, batch size:$\mathcal{K}$, $L_2$ regularization coefficient:$\lambda$.

**Output:** Predicted traffic matrix data:$Y_{t+1,t+2,t+3,\ldots,t+T}$.

1: **Initialize** GCN-GRU network with random weight:$\theta$.
2: **Input** data for standardization.
3: **Obtain** adjacency matrix A of SDWN network topology and calculate $\widehat{\mathrm{D}}^{-1/2}\widehat{\mathrm{A}}\widehat{\mathrm{D}}^{-1/2}$
4: **for** *episode* $= 1 \leftarrow \mathcal{M}$ **do**
5: Generate implicit H according to GCN-GRU network $H_{t+1,\ldots,t+T} = GCGRU(X_{t-n+1}, \widehat{\mathrm{D}}^{-1/2}\widehat{\mathrm{A}}\widehat{\mathrm{D}}^{-1/2})$
6: Calculate the final prediction result based on FC network $Y_{t+1,\ldots,t+T} = FC(X_{t+1,\ldots,t+T})$
7: Calculate loss function: $Loss = \frac{1}{\mathcal{K}}\sum_{i=1}^{\mathcal{K}}\left(Y_t - \widehat{Y}_t\right)^2 + \frac{1}{2}\lambda\|\theta\|_2^2 \rightarrow 0$, update model weights $\theta = \widehat{\theta}$
8: **end for**

Previous studies primarily focused on network traffic research using the TM [40] as the target, but the use of a single traffic tracking and statistical method proved to be costly and yielded limited results. To better adapt to the dynamic changes and spatiotemporal characteristics of wireless network traffic, this paper proposes a GCN-GRU-based network state prediction method that incorporates spatiotemporal feature correlation. This method aims to address the intricate and complex network traffic issues in SDWN and enhance the intelligent agent's perception ability for future time points in the TM.

As shown in Algorithm 1. we take the converted TM $X_{1,2,3\cdots,t}$, the adjacency matrix $A$ of the graph, the learning rate $\alpha$, the number of training episodes $\mathcal{M}$, the batch size $\mathcal{K}$, and the $L_2$ regular term coefficient $\lambda$ as inputs. On this basis, the algorithm will output the predicted TM $Y_{1,2,3\cdots,t}$. The first and second lines randomly initialize the network weights $\theta$ and standardize the input $X_{1,2,3\cdots,t}$. The third line calculates the degree matrix parameters $D$ used in the prediction network convolution from the adjacency matrix $A$. The fourth to eighth lines describe the iterative network training process. The GCN-GRU network generates the hidden layer parameters $H$. Finally, the output is produced by filtering the prediction results through a fully connected layer. During training, the weight parameters of the network are updated based on the minimum quantized true value and the square difference of the traffic prediction results. The predicted traffic matrix output by the GCN-GRU traffic prediction algorithm will serve as the input to the DRL algorithm.

### C. DESIGN OF DRL-PPO INTELLIGENT ROUTING ALGORITHM

The most important task when solving wireless network routing optimization problems through DRL algorithms is to design a state space, a reward function, and an action space for decision-making that match the current environment. The state space describes the current state that a DRL agent can observe from the environment. The agent perceives the current network state information, and its learning direction in updating its policy is then guided by the reward function $R$. After the agent executes the action $A_t$ in state $S_t$, the current state transitions to the next state $S_{t+1}$, and $A_t$ receives the corresponding reward value $R_{t+1}$ according to $S_{t+1}$. When the agent obtains the maximum reward value, the path chosen by the agent is the optimal path. The corresponding interaction process is shown in Fig. 5.

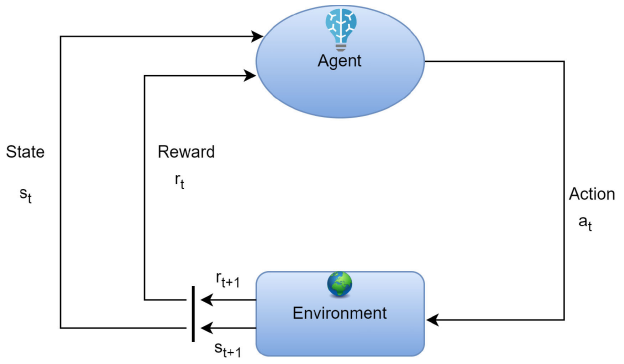



**FIGURE 5.** Interaction process between agent and environment in RL.

In the following, we provide a detailed introduction to the design of the state space, action space, and reward function in the DRL algorithm proposed in this article as well as the importance sampling method, gradient clipping optimization scheme, and PPO update strategy used in the PPO algorithm [42].

#### 1) STATE SPACE $\mathcal{S}$

The state space describes the current state of the network environment as perceived by the agent, which includes the current location state $S_l$ and the network state information $S_{info}$ of that location. The location state $S_l$ is composed of a two-dimensional matrix $M_l$ of size $H \times W$, as shown in Fig. 6. The main diagonal of this matrix represents the current effective location of the agent, which is specified as $M_l = [(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)]$, where $n$ represents the number of network topology nodes. The starting point is represented by *start*, the ending location is represented by *end*, and the remaining elements of the matrix are represented by 0, indicating location that the intelligent agent has not yet passed or will not pass. In the update process, the location information of the intelligent agent will be updated along with the currently selected action. When the next location state of the intelligent agent reaches the end position, the algorithm has completed the current iteration of routing decision-making.

In this algorithm, the current network state information $S_{info}$ is composed of multiple two-dimensional state matrices $M_{ij}$ that contain various link information, such as the

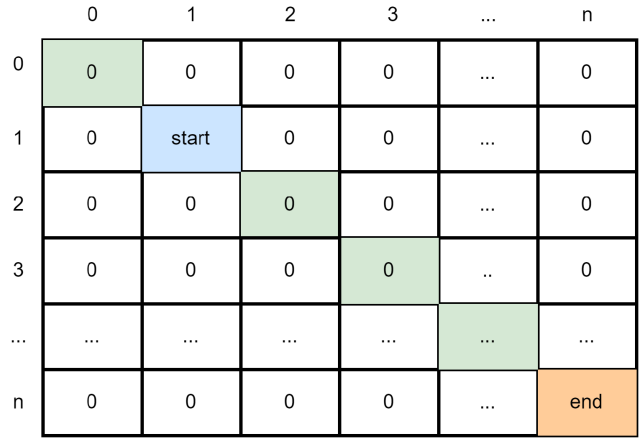



**FIGURE 6.** $M_l$ position state matrix.

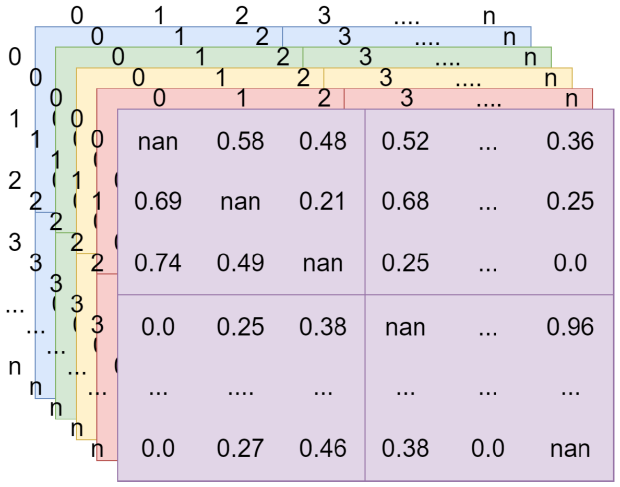



**FIGURE 7.** $M_{info}$ network information state matrix.

remaining bandwidth $bw_{free(ij)}$, the link delay $delay_{ij}$, the packet error rate $pkt_{err(ij)}$, the distance $distance_{ij}$ between wireless APs, and the packet loss rate $loss_{ij}$. The row number $i$ and column number $j$ of the matrix represent the identifying numbers of particular switch nodes; if $i \neq j$, the element in the $i$ row and $j$ column represents the link $e_{(i,j)}$ from node $i$ to node $j$; while if $i = j$, the element in the $i$ row and $j$ column represent node $i$ or $j$ itself. Here, $i, j \leq n$, and $e_{(i,j)} = e_{(j,i)} \in E$, where $E$ denotes the set of links in the network. The network state information matrix $M_{info}$ is shown in Fig. 7. In this matrix, because the elements on the main diagonal do not correspond to network information between two switches, they are assigned an infinite *nan* value. Notably, the network link data obtained in an SWDN environment are unfavorable for training neural network models through gradient descent because the gradient explosion phenomenon tends to occur during the training process due to their large numerical values; this phenomenon reduces the model convergence speed of the algorithm and has a significant impact on the search trajectory of the agent. Therefore, the min-max method [43] is used to normalize the flow matrix, limiting the values of the elements in the matrix to within the range $[a, b]$, where $a, b \in [0, 1]$. In Equation (8), $TM_{ij}$ represents the normalized flow matrix, $m_{ij}$ represents an element of the normalized flow matrix, as well as max(TM) and min(TM) represent the maximum and minimum values of the matrix elements, respectively. Due to the need for data preprocessing and the avoidance of calculation errors when the denominator is zero, a small numerical value is added to the denominator to ensure the accuracy of the calculation.

$$\overline{TM}_{ij} = a + \frac{\left(m_{ij} - \min\left(TM\right) \cdot (b - a)\right)}{\left(\max\left(TM\right) - \min\left(TM\right) + 1e^{-6}\right)} \tag{8}$$

During the training process, to meet the input requirements of convolutional neural networks in RL, the location state matrices $M_l$ and the network state information matrices $M_{info}$ need to be concatenated in the channel dimension $C$ to form the input state space matrices. Namely, the two-dimensional matrix $M_l$ and $M_{info}$ of size $H \times W$ for $C$ channels are transformed into a three-dimensional spatial matrix tensor $T_l$ and $T_{info}$ of size $H \times W \times C$, where each channel corresponds to a two-dimensional matrix of $H \times W$.

### 2) ACTION SPACE $\mathcal{A}$

The action space describes the set of actions among which an intelligent agent must select during its interaction with the environment. After the intelligent agent executes an action based on the current state, the current state transitions to the next state. The design of the state space in this article is composed of location information and network information to enable the agent to quickly find the optimal path in the environment, improve the interaction between the agent and the environment, and reduce the dimensionality of the possible actions in the action space. In this article, each next-hop adjacent node is considered as a possible action node in the DRL action space, which is expressed as $A = \{a_1, a_2, \ldots, a_i, a_{n-m}, a_n\} \in \{N_1, N_2, \ldots, N_i, N_{n-m}, N_n\}$. Here, $N_i$ represents a next-hop node, $m$ represents the number of invalid actions, $n$ represents the total number of wireless nodes, and $a_i$ represents the current action being executed. In the process of action selection, there are three possible situations of the intelligent agent that need to be considered to provide corresponding decision-making strategies:

- If the next selected action node is not an adjacent node of the current node, the selected node will not be added to the selected path matrix. The agent will remain in its current location, and the behavior of the agent will be punished;
- If the selected action node belongs to the set of adjacent nodes but will cause the paths in the path matrix to form a loop, the agent will also be punished to some extent. Nevertheless, the location state of the intelligent agent will be changed, and the link information matrix $T_{info}$ will be updated;

- When the agent selects the endpoint node, its location state will change, and a reward value should be returned to incentivize the intelligent agent to select this path, thus completing the current iteration of the routing decision-making process.

#### 3) REWARD FUNCTION $\mathcal{R}$

The reward function is used to optimize the direction of agent learning. In the algorithm designed in this article, the agent always makes routing decisions toward the direction of the maximum reward value, comprehensively considering the overall performance of the network to achieve multi-objective routing optimization. Therefore, the reward value is calculated using the remaining link bandwidth $bw_{free(ij)}$, the link delay $delay_{ij}$, the link packet error rate $pkt_{err(ij}$, the distance $distance_{ij}$ between wireless APs, and the link packet loss rate $loss_{ij}$ in the SDWN network. Accordingly, the reward value between two nodes, also called the reward value for the next hop, can be represented by a two-dimensional matrix $R_{link}$, and the final reward value for the entire link can be represented by a two-dimensional matrix $R_{total}$. The former allows the agent to make local routing decisions, while the latter allows the agent to make global routing decisions. The purpose of designing these two types of reward functions is to prevent the agent from falling into locally optimal solutions and enable the agent to quickly find the optimal path. The relationship between the two rewards is represented by Equation 9, where the row number $i$ and column number $j$ of the matrix represent the identifiers of particular switch nodes.

$$R_{total} = \sum_{i,j=0}^{n} R_{link(ij)} \tag{9}$$

During its interaction with the environment, an intelligent agent may encounter the following three situations, and the reward value for the next hop will be split into three different rewards accordingly. The intelligent agent will receive the corresponding reward value based on the current environmental situation for cumulative learning.

- When the agent selects an adjacent node of the current node as the next action, the path it passes through does not form a loop, and the next state is not the destination state, that is, $S_{t+1} \neq None$, the reward value obtained is $R_{link}$. This reward value is calculated as shown in Equation (10), where $\beta_1, \beta_2, \beta_3, \beta_4, \beta_5 \in [0, 1]$. The link performance indicators that will be prioritized for optimization can be controlled by adjusting the weight values $\beta$.

$$\begin{aligned} R_{link} = {} & \beta_1 \overline{bw}_{ij} - \beta_2 \overline{delay}_{ij} - \beta_3 \overline{pkt}_{err(ij)} \\ & - \beta_4 \overline{distance}_{ij} - \beta_5 \overline{loss}_{ij} \end{aligned} \tag{10}$$

- When the next action selected by the agent is an adjacent node of the current node and forms a loop in the path traveled by the agent, the reward value obtained is $R_{loop}$. This reward value is given in the Equation (11), where $\xi_1$ is the discount factor, the negative sign indicates that the agent is penalized with a negative value to discourage it from performing this action again, and $R_s$ is the standard reward; $\xi_1, R_s \in [0, 1]$. By varying the size of the discount factor to optimize the decision-making direction of the agent, the agent's motion trajectory can be made more stable, and the agent's routing policy can be dynamically adjusted according to different network topologies.

$$R_{loop} = -\xi_1 R_s \tag{11}$$

- When the agent selects an action node that is not adjacent to the current node, the reward value obtained is $R_{non\_edges}$. As shown in Equation (12), the standard reward $R_s$ is multiplied by a discount factor $\xi_2$, where $\xi_2, R_s \in [0, 1]$, and the negative value is taken to penalize the selection of invalid actions and increase the probability of the agent selecting correct actions.

$$R_{non\_edges} = -\xi_2 R_s \tag{12}$$

#### 4) IMPORTANCE SAMPLING

The general gradient descent update algorithm is used because the value estimated by the advantage evaluation function is not completely accurate, so there will be deviations in the data. If the data value obtained by the agent deviates too far from the actual value after the execution of a policy, the next sample will also strongly deviate from the estimate, resulting in the agent executing a policy that also strongly deviates from the predicted policy. At the same time, because of the different distribution of the data used for training, the network parameters $\theta$ will change to $\theta'$ after learning. Therefore, the importance sampling method is used in this paper to adjust the distribution of the data, avoid problems caused by data estimate deviation, and improve the efficiency of policy updates and sample utilization. Specifically, Equation (13) is importance sampling, where $w_t^{IS}$ is the weight of importance sampling, and $\pi_\theta(a_t|s_t)$ and $\pi_{\theta old}(a_t|s_t)$ represent the probabilities of taking action $a_t$ in state $s_t$ when using the new policy and the old policy, respectively.

$$w_t^{IS} = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \tag{13}$$

#### 5) GRADIENT CLIPPING OPTIMIZATION

During importance sampling, to avoid a large difference between the original data distribution $p(x)$ and the data distribution $q(x)$ that can be sampled, it is necessary to add a constraint $\sigma$ on the distance between the two, as represented by the KL divergence. Because the loss function of the PPO DRL algorithm is constrained by relevant conditions and calculated via the conjugate gradient method and because the KL divergence constraint, as an additional constraint condition, does not participate in the update of the convolutional neural network parameters, it is difficult to dynamically adjust the parameters in the KL divergence to adapt to different data distributions. Therefore, the KL divergence is introduced into the loss function as a regularization term for optimization.

The gradient descent formula with the KL divergence regularization term is shown in Equation (14).

$$\begin{aligned} J_{PPO}^{\theta^k} &= J^{\theta^k}(\theta) - \sigma KL\left(\theta, \theta^k\right) \\ J^{\theta^k}(\theta) &\approx \sum_{(s_t,a_t)} \frac{p_\theta(a_t|s_t)}{p_{\theta^k}(a_t|s_t)} A^{\theta^k}(s_t, a_t) \\ A^{\theta^k}(s_t, a_t) &= R_t + \gamma V(s_{t+1}) - V(s_t) \end{aligned} \tag{14}$$

where $\theta$ represents the policy parameters in the neural network; $\sigma$ represents the weight coefficient of the KL divergence, satisfying $\sigma \in [0, 1]$; $\theta^k$ represents the network parameters after $k$ iterations; $J^{\theta^k}$ represents the likelihood function of gradient decline after $k$ iterations; $p_\theta(a_t|s_t)$ represents the action-state transition probability distribution function; $A^{\theta^k}(s_t, a_t)$ represents the advantage evaluation function after $k$ iterations, which is used to evaluate the value of the current strategy of the agent; $R_t$ refers to the reward function at time $t$; $\gamma$ is a reward attenuation factor; and $V(s_t)$ and $V(s_{t+1})$ are the estimated values for the current state and the next state, respectively.

Due to the computational complexity of the KL divergence [44] formula and the difficulty of selecting an appropriate penalty factor $\sigma$ to adjust the similarity between the old and new policies, to better adapt to dynamic network topology changes and uneven data distributions while effectively solving the problem of gradient explosion or vanishing, this algorithm uses the gradient clipping method in place of the KL divergence. Gradient clipping not only effectively limits the policy update amplitude but also improves the model convergence speed and performance of the algorithm. The Equation for the gradient clipping method is given in (15).

$$\begin{aligned} clip_t(\theta) &= \min\left(\max\left(\eta_t(\theta), 1-\epsilon, 1+\epsilon\right)\right) \\ \eta_t(\theta) &= \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \end{aligned} \tag{15}$$

where $\theta$ represents the policy parameters in the neural network, $\epsilon \in [0, 1]$ is the clipping coefficient used to adjust the amplitude of the policy update, and $\eta_t(\theta)$ is the ratio of the new policy $\pi_\theta(a_t|s_t)$ to the old policy $\pi_{\theta old}(a_t|s_t)$ at time $t$.

Gradient clipping limits the ratio between the new policy function $\pi_\theta(a_t|s_t)$ and the old policy function $\pi_{\theta old}(a_t|s_t)$ to a range of $(1-\epsilon, 1+\epsilon)$. If the gradient norm exceeds this range, clipping is applied, and finally, the minimum value is taken as the output result.

#### 6) PPO UPDATE STRATEGY

The policy update formula of this algorithm maximizes the upper limit on the expected cumulative return while limiting the amplitude of the policy updates to ensure the convergence and stability of the algorithm. The policy update formula of the PPO algorithm is shown in Equation (16).

$$\theta_{t+1} = \operatorname{argmax}_\theta \hat{E}_t \left[ \min \left( \begin{array}{l} \eta_t(\theta) A^{\theta^k}(s_t, a_t), \\ \operatorname{clip}\left( \begin{array}{l} \eta_t(\theta), \\ 1-\epsilon, \\ 1+\epsilon \end{array} \right) A^{\theta^k}(s_t, a_t) \end{array} \right) \right] \tag{16}$$

where $\widehat{E}_t$ represents the expected value at time $t$, $\eta_t(\theta)$ is the ratio of the new and old policies at time $t$, and $A^{\theta^k}(s_t, a_t)$ is the advantage evaluation function for executing action $a_t$ after $k$ iterations, used to measure the quality of the executed policy. The function $clip(r_t(\theta), 1-\epsilon, 1+\epsilon)$ ensures that the update does not deviate too far from the old policy.

### D. DESCRIPTION OF THE DRL-PPONSA ALGORITHM

The implementation of the DRL-PPONSA algorithm framework is shown in Algorithm 2. Based on the input source node set $N_s$ and destination node set $N_d$, the optimal path from $N_s$ to $N_d$ is found from the currently observed network environment topology $G$. The prediction matrix output by Algorithm 1 is used as the network state information matrix that is also provided as input to Algorithm 2. In addition, the input to the algorithm includes the location state matrix $M_l$ corresponding to the current network state, the learning rate $\alpha$, the network parameter update frequency $f_{update}$, the size $\mathcal{K}$ of each batch drawn from the experience pool, the hyperparameter $\epsilon$ used in importance sampling, and the number of training episodes $\mathcal{M}$.

Lines 1-4 mainly initialize the parameters of the policy function $\pi_\theta(s)$ and the value function $V_\psi(s)$, while leaving the experience pool empty. In lines 7 and 8, the location state matrix $M_l$ and network state information matrix $M_{info}$ are reset based on a randomly selected source node $N_s$ and destination node $N_d$. Then, the two matrices are concatenated in the channel dimension to form matrices $T_l$ and $T_{info}$ with dimensions of $N \times N \times C \times \mathcal{K}$ as the initial state $S_t$, where $n$ is the number of nodes, that is, the size of the action space, and $\mathcal{K}$ is the batch size. Then, training is started in environments with different remaining bandwidths, delays, packet loss rates, packet error rates, and link transmission distances. Lines 5-35 describe one iteration of the training process, in which the agent travels from the source node to the destination node and outputs the selected path.

From line 9 to line 14, the agent executes the policy function $\pi_\theta(s)$ to obtain the current state matrix $s_t$, the selected action set $a_t$, and the probability $p(a_t)$ of selecting each action as well as the total cumulative reward value $r_t$ in the current environment. The advantage function $\widehat{E}_t$ is then evaluated, and based on the reward, the action policy is updated. Finally, the obtained next state value, current state value, and cumulative reward are stored in the experience pool to be used to train the parameters of the neural network.

Lines 15 to 22 describe the process of updating the parameters of the algorithm, mainly consisting of the loss function computation and update processes for the critic and actor

networks. First, relevant data are retrieved from the experience pool, including the current state matrix $s_t$, the selected action and its corresponding probability $p(a_t)$ from the policy distribution, the cumulative reward value $r_t$ of the agent, and information regarding the state matrix $s_{t+1}$ that the agent updates. The previously initialized advantage function $\widehat{E}_t$ is updated, and then the loss function of the policy network is updated based on the next-action probability $p(a_{t+1})$ and the previous-action probability $p(a_t)$, using the likelihood function $J^{\theta^k}_{PPO}(\theta)$ to update the network weights. Finally, the cumulative reward values are used to calculate the loss value for evaluating the network, guiding the agent to prioritize the path with the maximum cumulative reward.

At the end of lines 23 to 38, it is judged whether the agent has reached the destination. If the next state is not the destination, the cumulative reward value is calculated. The calculated reward value is the discounted link reward. If the agent has reached the destination, the target reward value is received. At the same time, the parameters of the two networks are updated in accordance with the likelihood function. When the agent reaches the destination, the current iteration cycle ends, and the status of the intelligent agent is updated to output the currently selected path. After the path is updated, a reward $r_t$ and a new network state $s_{t+1}$ will be obtained. In this way, the performance of the network is optimized toward the maximum reward value until the optimal routing policy is found.

**Algorithm 2** DRL-PPONSA

**Input:** location information matrix $M_l$, network status information matrix $M_{info}$, source nodes set $N_s$, destination nodes set $N_d$, learning rate $\alpha$, n-step $n_{step}$, batch-size $\mathcal{K}$, clip-param $\epsilon$, ppo-update-time $n_{update}$, training episodes $\mathcal{M}$.
**Output:** Intelligent path information for $(N_s, N_d)$.

**Initialize** actor network with random weight $\theta$.
**Initialize** critic network with random weight $\psi$.
**Initialize** PPO policy function $\pi_\theta(s)$ and value function $V_\psi(s)$.
**Initialize** n-steps buffer-capacity $B$.
**for** $episode = 1 \leftarrow \mathcal{M}$ **do**
  **for** $M_{bw_{free}}, M_{delay}, M_{loss}, M_{pkt_{error}}, M_{distance}$ in Network Information Storage **do**
    Reset enviroment with $(N_s, N_d)$.
    Get $s_t \leftarrow stack(M_l, M_{info}) \leftarrow (T_l, T_{info})$. //concatenate in the channel dimension
    **while** True **do**
      Run policy $\pi_\theta$ for $n_{update}$, collecting $(s_t, a_t, r_t, p(a_t))$
      Estimate advantages $\widehat{A}_t = \sum_{t'>t} \gamma^{t'-t} r_{t'} - V_\phi(s_t)$;
      Update the strategy of the action $\pi_{\text{old}} \longleftarrow \pi_\theta$;
      Execute action $a_t$ and observe reward $r_t$ and next state $s_{t+1}$;
      Store $(s_t, a_t, r_t, p(a_t), s_{t+1})$ in $B$.
      **for** $i = 1 \leftarrow n_{update}$ **do**
        **for** index in $(range(len(\mathcal{K})), \mathcal{K}, False)$ **do**
          Sample minibatch $(s_i, a_i, r_{i:i+n}, s_{i+n})$ and get $p(a_t)$ from Transition
          Update the advantages $\widehat{A}_t$ and calculate the qualifying ratio $r = \frac{p(a_t)}{p(a_{t+1})}$
          Calculate the likelihood function of gradient descent $J^{\theta^k}_{\text{PPO-Clip}}$ //The Equation is given by (14)
          Update $\theta$ by a gradient method w.r.t $J^{\theta^k}_{\text{PPO-Clip}}$
          Calculate the actor loss and critic loss.
        **end for**
        **if** $s_{i+n}$ is not None **then**
          $R \leftarrow R_{i:i+n} + \gamma \sum_{t=1}^{T} R_{link(t)}$
        **else**
          $R \leftarrow R_{i:i+n}$
        **end if**
        Update policy network parameters $\theta \leftarrow \theta + \zeta_\theta \nabla J^{\theta^k}_{\text{PPO-Clip}}(\theta)$
        Update critic network parameters $\psi \leftarrow \psi + \zeta_\psi \nabla J^{\psi^k}_{\text{PPO}}(\psi)$
      **end for**
      **if** The agent arrives at the destination node **then**
        Break
      **end if**
      $s_t \leftarrow s_{t+1}$
    **end while**
    Output the path selected by the agent from $(N_s, N_d)$
  **end for**
**end for**

## V. EXPERIMENTAL SETTINGS AND PERFORMANCE EVALUATION

This section primarily evaluates the DRL-PPONSA algorithm. First, the parameter settings of the simulation environment and the method of traffic generation are described. Second, the performance of the prediction algorithm and the parameter settings of the DRL algorithm are analyzed. Furthermore, performance comparisons are made between the proposed algorithm and the Dueling Deep Q-Network (DQN), OSPF, DVRP, and link state routing protocol (LSRP) algorithms. Finally, a comprehensive analysis of the results is presented.

### A. SIMULATION ENVIRONMENT SETTINGS

The control plane used in these experiments uses Ryu as the SDWN controller, which is responsible for matching and issuing flow table entries and responding to events. For the data plane, the Mininet-WiFi 2.3.1b [50] simulation environment is used to build a simulated wireless network topology, and the sFlow-RT tool [45] is used to monitor the flow situation in the network. The entire simulation environment is deployed on an Ubuntu 18.04.6 server equipped with a GeForce RTX 3090 graphics card. Ryu [48] uses the southbound interface of the OpenFlow 1.3 protocol to communicate with Mininet-WiFi's Open vSwitches [46]. The dataset is collected by Ryu and stored as a pickle file in graph format. Finally, the interaction between SDWN and DRL is achieved using Python 3.8 and Pytorch 1.11.0. The simulation tools are listed in Table 2.

The experimental topology built using the Mininet-WiFi simulation platform is shown in Fig. 8. It is a simulated wireless IoT data center topology that includes 14 nodes

**TABLE 2.** Simulation tools.

| Tools | Version | Function |
|---|---|---|
| Mininet-WiFi | 2.3.1b | network topology construction |
| Ryu | 4.3.4 | flow table distribution |
| GPU | GeForce RTX3090 | accelerated calculation |
| Ubuntu | 18.04.6 | experimental system |
| sFlow-RT | 3.0 | flow monitoring |

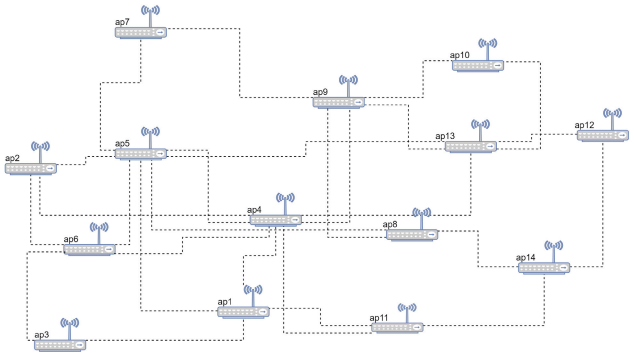



**FIGURE 8.** The wireless network topology of APs.

and 25 links. The dashed lines in this figure represent wireless links between wireless APs, and each wireless AP is connected to a STA. Table 3 provides information such as the simulation range, simulation time, and basic parameters of the simulation environment between the wireless APs and STAs. The link parameters between wireless APs are randomly generated following uniform distributions. The bandwidth is set to $5 \sim 40Mbps$, the delay is $1 \sim 10ms$, the link packet loss rate is set to $0.1 \sim 1\%$, and the distance between wireless APs is set to $30 \sim 110$ meters.

To simulate real wireless network traffic usage, we use the iPerf3 [49] tool to write Python scripts that generate traffic information. A script sends User Datagram Protocol (UDP) traffic requests to a server through a client, randomly selecting the client and server for multi-objective traffic and adjusting the size of the traffic based on the gravity model [48]. The flow size is controlled within the range of $0 \sim 50MHz$ to ensure that the flow information reaches its peak during the 10:00-15:00 period of the day and slowly decreases during other periods. The traffic information collection interval is once every 5 s. Finally, the Ryu controller monitoring module script generates 1000 instances of graph data for 25 traffic matrices between the 14 nodes and writes the graph data to a pickle file. The elements in the matrices include link information such as residual bandwidth, used bandwidth, delay, packet loss rate, and distance. The sFlow-RT tool [51] is used to monitor the average traffic flow information in bits per second, as shown in Fig. 9.

**TABLE 3.** Simulation parameters.

| Parameters | Value | Parameters | Value |
|---|---|---|---|
| MAC protocol | IEEE 802.11g | Frequency band | $2.4GHz$ |
| Number of APs | 14(nodes) | Number of STAs | 14(nodes) |
| CCA threshold | $-62dBm$ | Channel | $1, 6$ |
| Propagation loss | log-distance | Transmission power | $21dBm$ |
| Receiving gain | $5dBm$ | Transmitting gain | $5dBm$ |
| Signal range | $38 \sim 140m$ | Simulation area | $300x300m^2$ |
| Modulation technique | OFDM | Simulation time | $180min$ |

### B. ANALYSIS OF GCN-GRU PREDICTION NETWORK PERFORMANCE AND DRL PARAMETERS

The GCN-GRU model fully utilizes the capabilities of GCNs to obtain spatial graph information and utilizes the advantages of GRU networks for time series modeling to mine hidden and unknown traffic information from SDWN networks, thereby improving the perception ability of the DRL algorithm and making the generated routing policies more forward-looking and robust. As shown in Fig. 10(a), the reward values obtained by an agent using the GCN-GRU prediction mechanism are significantly higher than those obtained without the prediction mechanism. Moreover Fig. 10(b) show that the number of steps taken by the agent using the prediction mechanism is significantly fewer than that taken by the agent without the prediction mechanism. This is because the prediction mechanism can capture unknown traffic status information in the network, allowing the intelligent agent to perceive more traffic information, make judgments in advance for routing decisions, and avoid selecting congested nodes. Moreover, when relying solely on Ryu's network measurement mechanism, it is difficult to detect hidden traffic information in the SDWN network, and it is also difficult to adjust policies promptly to adapt to the distribution of traffic. Therefore, the proposed traffic prediction algorithm based on the GCN-GRU model can enhance the exploration space of the DRL-PPO intelligent routing algorithm to optimize the action selection and increase the reward value, thus yielding more appropriate routing policies for SDWN network flows and improving the utilization of network resources. These findings verify that the proposed prediction mechanism can improve the performance of intelligent routing algorithms.

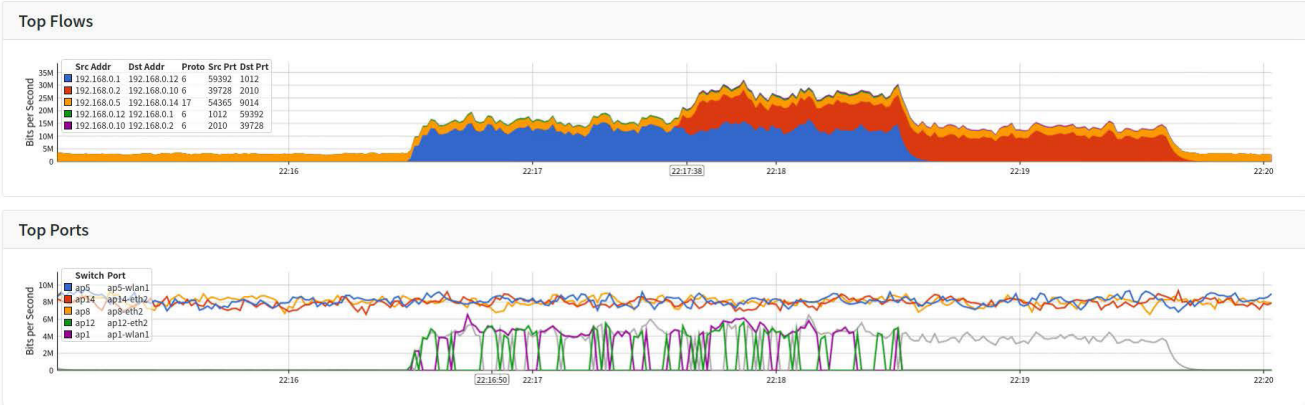



**FIGURE 9.** The curve of port traffic information versus time.

The intelligent routing algorithm designed in this article adopts the PPO algorithm, with updates based on policy gradient descent. It uses an online learning strategy, in which the same agent is used in both the training stage and the policy execution stage. During the learning process, the intelligent agent can complete policy selection for the next state, and the selection for the previous state does not affect the policy selection for the next state, while the opposite is true in offline learning. The basic parameters set for the intelligent agent during the training process are shown in Table 4. For PPO DRL, a four-layer network structure is adopted, which includes a two-layer actor network and a two-layer critic network. The actor network is used to sample the probability of the current action and the probability of the next state's action, while the critic network is used to evaluate the cumulative reward of the current state and the cumulative reward of the next state. For the optimizer, adaptive moment estimation (Adam) is used, which can adaptively update its learning rate and reduce the consumption of computer memory, enabling the network model of the intelligent routing algorithm to quickly converge during the training process.

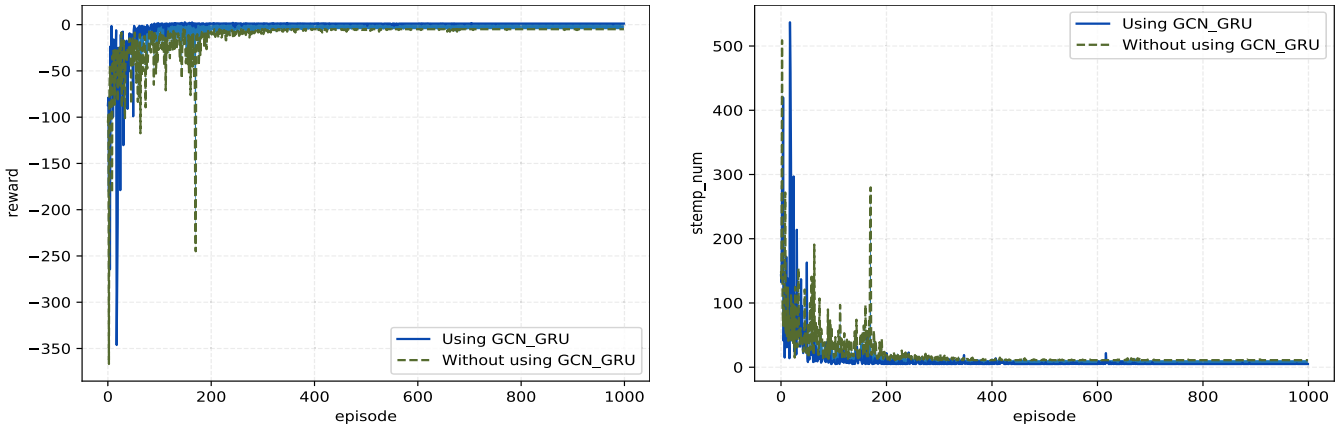
Choosing appropriate weight factors will improve the model convergence effect of the intelligent routing algorithm and accelerate the convergence process. The routing algorithm in this article is an intelligent routing algorithm aimed at optimizing the parameters of multiple target links, which includes maximizing the remaining bandwidth and minimizing the delay, packet loss rate, packet error rate, and transmission distance. In the experimental process, we choose the default weights $[0.7, 0.3, 0.1, 0.1, 0.1]$ for the optimization objective, which means that $\beta_1 = 0.7$, $\beta_2 = 0.3$, $\beta_3 = 0.1$, $\beta_4 = 0.1$, and $\beta_5 = 0.1$, because these weight settings do not adversely affect the convergence of the algorithm. These weights control the proportional contributions of different types of link information during intelligent routing optimization. For example, the weight with the highest value is $\beta_1$, which means that when selecting a route, the main consideration will be to maximize the remaining bandwidth, while other link information will be used as secondary

**TABLE 4.** DRL parameter settings.

| Parameter | Symbol | Value |
|---|---|---|
| Learning rates | $\alpha_1, \alpha_2$ | $1e^{-3}, 3e^{-3}$ |
| Bach size | $\mathcal{K}$ | 32 |
| Clip parameter | $\epsilon$ | 0.2 |
| PPO-update-time | $f$ | 10 |
| Buffer-capacity | $B$ | 3000 |
| Reward attenuation | $\gamma$ | 0.99 |
| Discount factors | $\xi_1, \xi_2$ | 0.5, 0.8 |
| episodes | $\mathcal{M}$ | 1000 |

optimization conditions to affect the decision-making of the agent. This multi-objective optimization approach is adopted to overcome the shortcomings of single-objective optimization because it is better to comprehensively consider the current state of SDWN network traffic to improve the utilization of network resources. Meanwhile, the setting of the reward discount factors and the setting of the hyperparameters of the deep learning network will directly affect the convergence and stability of the intelligent algorithm; therefore, a related parameter analysis is given below.

The setting of the reward and penalty weights, $\xi_1$ and $\xi_2$, respectively, can have an impact on the direction of the agent's decisions and the convergence efficiency of the algorithm. Improper values for $\xi_1$ and $\xi_2$ can prevent the agent from learning the optimal path, which can affect the convergence of the algorithm and even lead to non-convergence. As shown in Fig. 11(a), through multiple experiments, it has been found that setting the standard reward $R_s$ to 1 and $\xi_1 : \xi_2 = 0.5 : 0.8$ results in the fastest convergence speed with relatively stable waveforms, while $\xi_1 : \xi_2 = 0.5 : 1.5$ results




(a) The curve of reward versus episode

(b) The curve of step number versus episode

**FIGURE 10.** Comparison of the reward and step change curves with and without the implementation of the GCN-GRU prediction mechanism.

**TABLE 5.** The impact of the learning rate on algorithm convergence.

| $\alpha_1, \alpha_2$ | Reward | Steps | Episodes to reach convergence |
|---|---|---|---|
| $1e^{-1}, 3e^{-1}$ | $-0.42$ | 45 | 800 |
| $1e^{-2}, 3e^{-2}$ | $-0.82$ | 34 | 290 |
| $1e^{-3}, 3e^{-3}$ | 1.78 | 5 | 172 |
| $1e^{-4}, 3e^{-4}$ | $-180$ | 478 | 890 |
| $1e^{-5}, 3e^{-5}$ | $--$ | $--$ | $--$ |
| $1e^{-6}, 3e^{-6}$ | $--$ | $--$ | $--$ |

in the slowest convergence speed, taking nearly 400 iterations to reach convergence.

The following analysis focuses on the impact of the algorithm hyperparameters on the reward received by the intelligent agent. The following hyperparameters are analyzed: the learning rate $\alpha$, the batch size $\mathcal{K}$, and the PPO network update frequency $f$. The value of $\alpha$ determines when the objective function can converge to a local minimum and when it will converge to the global minimum. A suitable learning rate can make the objective function converge to a local minimum in a suitable amount of time. Since the PPO convolutional neural network consists of an actor network and a critic network, two different learning rates are set to adapt to the different network parameters and loss functions in the framework, namely, $\alpha_1$ and $\alpha_2$. As shown in Table 5, when $\alpha_1 = 1e^{-3}$ and $\alpha_2 = 3e^{-3}$, the agent obtains the maximum reward, and the convergence effect is the best. At the same time, the agent takes the fewest steps when making routing decisions and seeks the globally optimal routing policy.

The update frequency of the parameters in the PPO network affects the decision efficiency of the actor and critic networks. A suitable update frequency can enable the agent to make more accurate decisions and obtain higher reward values. In this experiment, we set the frequency $f$ to 1, 5, 10, and 15. As shown in Fig. 11(b), when the update frequency is set to 15, the agent obtains the maximum reward value, and the convergence effect is the best. Convergence is reached at approximately 170 iterations. However, when the update frequency is set to 1, the waveform exhibits large-amplitude jitter, which prevents the algorithm from converging.

The batch size $\mathcal{K}$ determines the number of samples sent to the neural network at one time, i.e., the amount of data that the agent retrieves from the experience pool. An appropriate batch size can make full use of the GPU's parallel computing power, thereby improving the training efficiency of the algorithm and reducing the time required for model training. When the $\mathcal{K}$ value is changed during training, the environment of the intelligent agent needs to be reinitialized to avoid the effects of the differences before and after this change. In this study, batch sizes of 16, 32, 64, and 128 are tested. As shown in Fig. 11(c), when the batch size is $\mathcal{K} = 16$ or $\mathcal{K} = 32$, the agent obtains the highest reward values, and the convergence speed is the fastest, with a relatively stable convergence effect and small fluctuations. The agent can obtain the maximum reward value within 200 episodes. However, when the batch size is $\mathcal{K} = 128$, the agent takes approximately 700 episodes to converge, and the fluctuations are significant. Therefore, the batch size $\mathcal{K}$ is set to 32 in this paper to analyze the impact of the PPO-KL and PPO-Clip methods on the executed routing policy of the intelligent agent in importance sampling.

As shown in Fig. 11(d), when using the PPO-Clip optimization solution, the agent can converge faster and obtain the maximum reward value, while the convergence waveform is relatively stable. When using the PPO-KL optimization solution, the waveform shows obvious fluctuations due to the sensitivity of the KL divergence weight $\sigma$ to the data distribution and the difficulty of setting its value. For PPO-Clip, this restriction does not need to be considered. Therefore, using

the PPO-Clip method can improve the performance of the algorithm.

### C. PERFORMANCE ANALYSIS

First, this subsection compares the convergence and stability of the proposed DRL-PPONSA algorithm with those of the value-based Dueling DQN algorithm. Next, it presents performance comparisons between the proposed algorithm and other classical and traditional algorithms. Finally, the experimental results are analyzed.

The policy-based approach shows better convergence and is more effective for problems with high-dimensional and continuous action spaces. It not only solves the challenges of model building for dynamic and diverse networks but also addresses the slow convergence and experience pool storage space issues of value-based RL. To avoid any confounding influence of the prediction mechanism, this section analyzes experiments conducted using the predicted traffic matrix while keeping other parameter settings as consistent as possible. As shown in Fig. 12, the DRL-PPONSA algorithm reaches convergence at approximately 170 iterations, while the Dueling DQN algorithm takes approximately 300 iterations to converge; thus, DRL-PPONSA shows an overall improvement of 43.33% in convergence speed. In the DRL-PPONSA algorithm, the prioritized experience replay scheme in Dueling DQN is replaced with the importance sampling technique. Importance sampling allows online training to be performed without storing the current sampled data; the entire experience pool is immediately deleted after the current data are used. In contrast, prioritized experience replay requires storing the data in the experience pool, and only some of those data are deleted when the capacity of the pool is exceeded, as this scheme relies on the extraction of historical experience data for offline training. Therefore, DRL-PPONSA can save computer memory, thus improving the convergence speed of the algorithm.

Next, the DRL-PPONSA algorithm is compared with the classical Dueling DQN algorithm and the traditional OSPF, DVRP, and LSRP algorithms. The main performance indicators used to evaluate the algorithms include the average network throughput $\overline{T}_{ij}$, the average network delay $\overline{delay}_{ij}$, the average network packet loss rate $\overline{loss}_{ij}$, the average network packet error rate $\overline{pkt}_{err(ij)}$, and the average distance $\overline{distance}_{ij}$ of wireless access points. Equation (17) is used to calculate the network throughput, where $e_{ij} \in E$ represents one of the edges in the link and $b_{e_{ij}}$ represents the current amount of data sent on that link. Equations (18)-(22) gives the corresponding expressions for average network throughput, average network delay, and other metrics.

$$T_{ij} = \sum_{i,j=0}^{n} \times \frac{bw_{use(e_{ij})} \cdot \sqrt{\left(1 - \left(loss_{e_{ij}} + pkt_{err(e_{ij})}\right)\right)} \cdot b_{e_{ij}}}{2 \cdot delay_{e_{ij}}} \tag{17}$$

$$\overline{T}_{ij} = \frac{1}{n_p} \sum_{k=1}^{n_p} \sum_{e_{ij} \in p_{sd}} T_{ij}. \tag{18}$$

$$\overline{delay}_{ij} = \frac{1}{n_p} \sum_{k=1}^{n_p} \sum_{e_{ij} \in p_{sd}} delay_{ij} \tag{19}$$

$$\overline{loss}_{ij} = \frac{1}{n_p} \sum_{k=1}^{n_p} \sum_{e_{ij} \in p_{sd}} loss_{ij} \tag{20}$$

$$\overline{pkt}_{err(ij)} = \frac{1}{n_p} \sum_{k=1}^{n_p} \sum_{e_{ij} \in p_{sd}} pkt_{err(ij)} \tag{21}$$

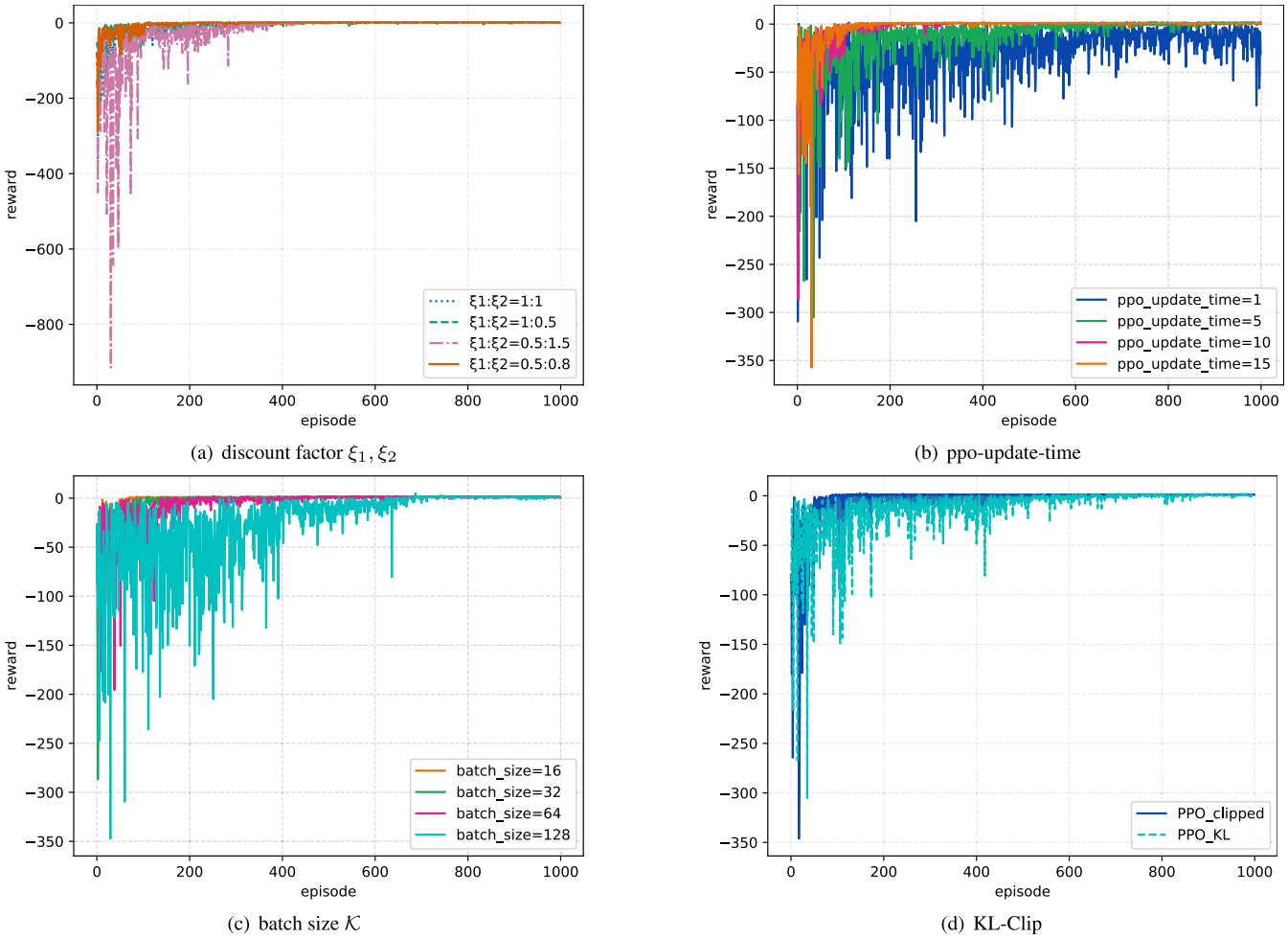
$$\overline{distance}_{ij} = \frac{1}{n_p} \sum_{k=1}^{n_p} \sum_{e_{ij} \in P_{sd}} distance_{ij} \tag{22}$$

Here, $p_{sd}$ represents the path from source node $s \in \{s_1, s_2, \cdots, s_n\}$ to destination node $d \in \{d_1, d_2, \cdots, d_n\}$, $n_p$ represents the number of pickles used to represent the amount of global network traffic obtained at time $t$.The traffic situation is collected for one day, and the average values are taken every three hours as the statistical results. The corresponding values are shown in the form of bar charts, and the experimental results are compared as follows.

The performance indicator shown in Fig. 13(a) measures the average total link throughput along the agent's path from the source node to the destination node. The DRL-PPONSA algorithm proposed in this paper achieves an average throughput that is 1.94% higher than that of the Dueling DQN algorithm, 31.04% higher than that of the OSPF algorithm, 47.81% higher than that of the DVRP algorithm, and 36.72% higher than that of the LSRP algorithm. The percentage improvements over the four compared algorithms, when the proposed algorithm performs best optimally, are 17.83%, 47.66%, 56.68%, and 48.76%, respectively. These findings indicate that the paths selected by the DRL-PPONSA algorithm have higher throughput, enabling the transmission of more data and better meeting the performance requirements for data transmission.

The measurement indicator in Fig. 13(b) is the average delay on the agent's path from the source node to the destination node. The delay achieved with the DRL-PPONSA algorithm is superior to $Dueling\ DQN_{delay}$, $OSPF_{delay}$, $DVRP_{delay}$, $LSRP_{delay}$. Specifically, the average delay under DRL-PPONSA is 6.08%, 39.58%, 61.48%, and 53.26% lower, respectively, than the average delays under the other four algorithms. The average link delay results indicate that compared to the other algorithms, our algorithm is more inclined to search for paths with lower network latency, thereby avoiding network congestion and meeting the performance requirements of low-latency networks.

The measurement indicator in Fig. 13(c) is the average packet loss rate on the agent's path from the source node to the destination node. Compared to the average packet loss rates $Dueling\ DQN_{loss}$, $OSPF_{loss}$, $DVRP_{loss}$, $LSRP_{loss}$, and $LSRP_{loss}$, the average packet loss rate under DRL-PPONSA




(a) discount factor $\xi_1, \xi_2$

(b) ppo-update-time

(c) batch size $\mathcal{K}$

(d) KL-Clip

**FIGURE 11.** Different parameter settings.

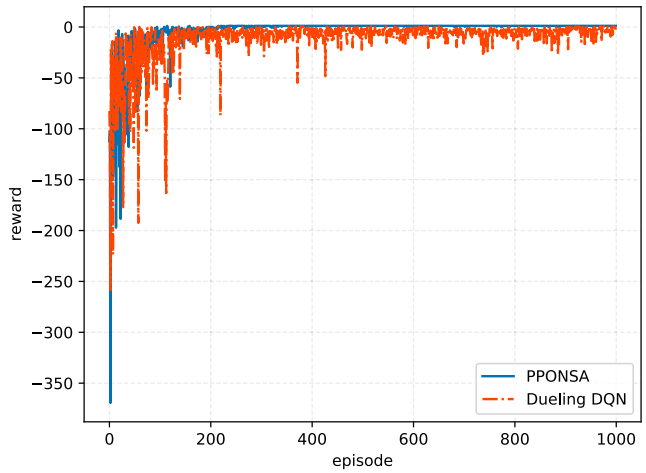
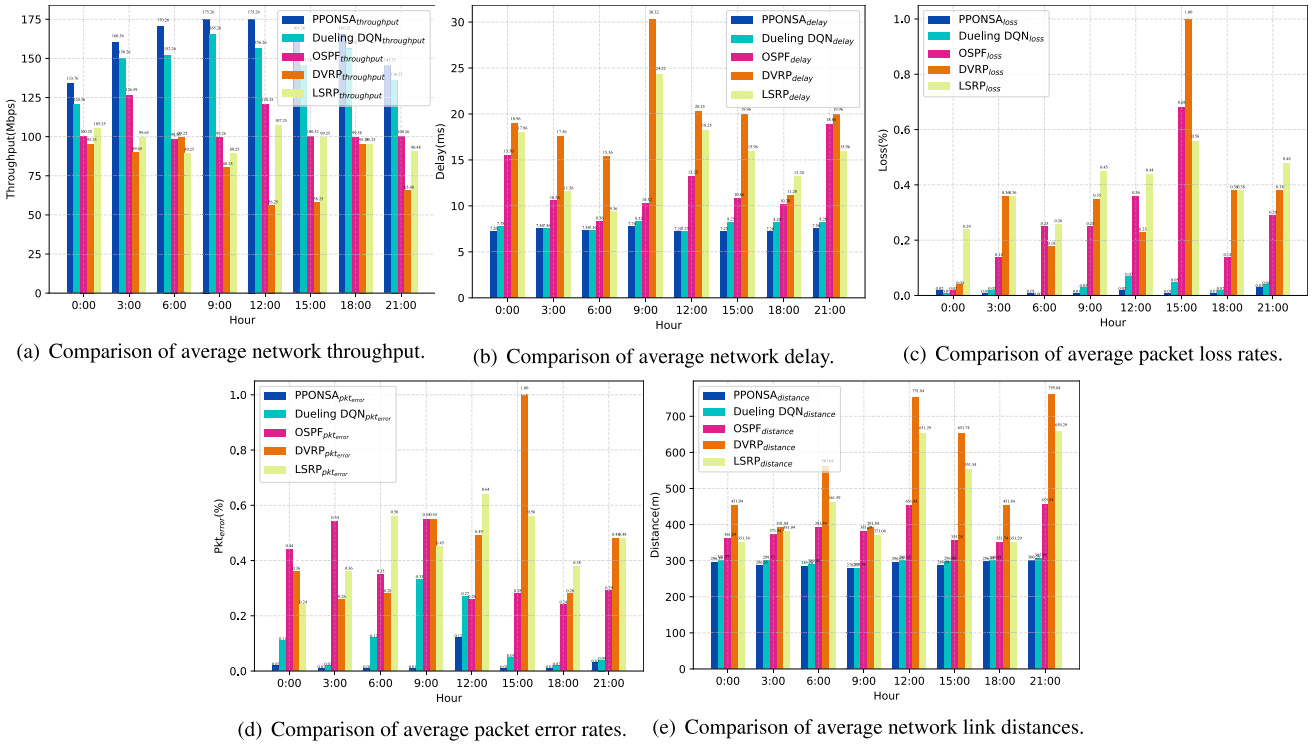



**FIGURE 12.** Comparison of convergence speed between PPONSA and Dueling DQN.

is reduced by 50.84%, 72.09%, 89.79%, and 82.07%, respectively; thus, it is significantly lower than the packet loss rates of the other algorithms. In this experiment, the packet loss rate on each link is set between 0.1% and 1%, so each link will experience packet loss with a certain probability. When a link is congested, the probability of packet loss increases. From the experimental results, it can be seen that the link throughput on the path selected by the algorithm proposed in this paper is higher than that on the paths selected by the other algorithms, indicating that the proposed algorithm can meet the transmission needs of a high-traffic network while effectively avoiding packet loss.

The measurement indicator in Fig. 13(d) is the average packet error rate on the agent's path from the source node to the destination node. The DRL-PPONSA algorithm has an average packet error rate that is 77.08%, 92.54%, 89.79%, and 94.05% lower than *Dueling DQN*$_{pkt_{err}}$, *OSPF*$_{pkt_{err}}$, *DVRP*$_{pkt_{err}}$, and *LSRP*$_{pkt_{err}}$, respectively, significantly lower than those of the other algorithms. When a link experiences certain congestion, the probability of erroneous packets will increase. From the experimental results, it can be seen that the link delay and packet loss rate on the path selected by the algorithm in this paper are both smaller than those of the other algorithms, indicating that the proposed algorithm can meet the transmission needs of high-traffic networks while effectively avoiding packet errors.




(a) Comparison of average network throughput.
(b) Comparison of average network delay.
(c) Comparison of average packet loss rates.
(d) Comparison of average packet error rates.
(e) Comparison of average network link distances.

**FIGURE 13.** **Comparison of performance with other algorithms.**

The metric shown in Fig. 13(e) is the average link distance on the agent's path from the source node to the destination node. The DRL-PPONSA algorithm achieves average reductions of 19.68%, 25.49%, 47.75%, and 38.43% in link distance compared to *Dueling DQN*$_{distance}$, *OSPF*$_{distance}$, *DVRP*$_{distance}$, and *LSRP*$_{distance}$, respectively. Although the average link distance of the proposed algorithm is similar to *Dueling DQN*$_{distance}$, it can find markedly shorter transmission distances than the traditional algorithms can. In the optimization process, the distance between wireless APs is used as one of the measurement indicators to enable the intelligent agent to find shorter paths for data forwarding, thereby avoiding the selection of redundant paths and reducing the consumption of network bandwidth resources.

The experimental results show that the intelligent agent of the DRL-PPONSA algorithm proposed in this paper can search for the optimal routing policy based on the designed reward function. The convergence speed is significantly better than that of the Dueling DQN algorithm, and the average throughput is significantly higher than those of the traditional OSPF, DVRP, and LSRP algorithms. Meanwhile, the packet loss rate and transmission distance are significantly lower than those of the OSPF, DVRP, and LSRP algorithms. Although the proposed algorithm has shortcomings in terms of latency, it is still better than the traditional algorithms in the case of link congestion. The algorithm proposed in this paper is more stable in terms of transmission delay and more suitable for a network environment with small delay jitter. In environments with different volumes of network traffic, intelligent agents can adjust their reward functions based on multiple optimization objectives to dynamically update their routing policies to effectively ensure that the specified performance requirements are met during data transmission. The above experimental comparisons confirm that the SDWN-based intelligent routing algorithm proposed in this paper, DRL-PPONSA, not only shows good convergence but also offers good performance and stability.

## VI. CONCLUSION

This paper presents a novel method named DRL-PPONSA for constructing intelligent routing paths in SDWN networks using DRL. A GCN-GRU prediction model is also incorporated to explore unknown traffic information in the network, allowing the controller to perceive the network situation in real time and achieve intelligent network control. In dynamic SDWN environments, DRL-PPONSA utilizes the policy gradient-based PPO RL method to enable an intelligent agent to construct routing paths with higher bandwidths, shorter transmission distances, lower delays, and lower packet loss and error rates based on the measured network parameters. Moreover, this agent can intelligently adjust the routing paths and perform flow table installation when the network parameters change. Comparisons with existing routing methods show that the proposed intelligent algorithm exhibits good convergence and stability, significantly enhancing the overall performance and service quality of wireless networks.

Furthermore, the proposed algorithm is effective in addressing the dynamic network routing optimization

problem. However, for large-scale and complex dynamic networks, relying on a single controller is insufficient to meet the performance requirements of current networks. Therefore, in future work, the integration of a multi-controller-based multi-agent routing optimization method will be investigated as a possible means of achieving a more efficient solution for SDWN routing.

## ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their helpful comments.

## REFERENCES

[1] Q. Yan, F. R. Yu, Q. Gong, and J. Li, "Software-defined networking (SDN) and distributed denial of service (DDoS) attacks in cloud computing environments: A survey, some research issues, and challenges," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 602–622, 1st Quart., 2016.

[2] F. Ning, L. Xue, and C. Xiaohui, "Implementation and evaluation of cooperative routing in software defined wireless networking," *J. Comput. Res. Develop.*, vol. 56, no. 5, pp. 967–976, 2019.

[3] C. Zhao, M. Ye, X. Xue, J. Lv, Q. Jiang, and Y. Wang, "DRL-M4MR: An intelligent multicast routing approach based on DQN deep reinforcement learning in SDN," *Phys. Commun.*, vol. 55, Dec. 2022, Art. no. 101919.

[4] M. Cicioğlu and A. Çalhan, "MLaR: Machine-learning-assisted centralized link-state routing in software-defined-based wireless networks," *Neural Comput. Appl.*, vol. 35, no. 7, pp. 5409–5420, Mar. 2023.

[5] X. Xiang, Y. Tian, X. Zhang, J. Xiao, and Y. Jin, "A pairwise proximity learning-based ant colony algorithm for dynamic vehicle routing problems," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 6, pp. 5275–5286, Jun. 2022.

[6] A. Manzoor, M. Hussain, and S. Mehrban, "Performance analysis and route optimization: Redistribution between EIGRP, OSPF & BGP routing protocols," *Comput. Standards Interfaces*, vol. 68, Feb. 2020, Art. no. 103391.

[7] M. Dogan and J. Lund, "Evaluating solar PV effects on California's hydropower generation with a hybrid LP-NLP optimization model," in *Proc. AGU Fall Meeting Abstracts*, vol. 2018, Dec. 2018, p. H31N-2141.

[8] L. Huang, M. Ye, X. Xue, Y. Wang, H. Qiu, and X. Deng, "Intelligent routing method based on Dueling DQN reinforcement learning and network traffic state prediction in SDN," *Wireless Netw.*, vol. 2022, pp. 1–19, Jul. 2022.

[9] M. Botvinick, S. Ritter, J. X. Wang, Z. Kurth-Nelson, C. Blundell, and D. Hassabis, "Reinforcement learning, fast and slow," *Trends Cognit. Sci.*, vol. 23, no. 5, pp. 408–422, May 2019.

[10] L. Ge, Y. Li, Y. Li, J. Yan, and Y. Sun, "Smart distribution network situation awareness for high-quality operation and maintenance: A brief review," *Energies*, vol. 15, no. 3, p. 828, Jan. 2022.

[11] L. Yang, X. Gu, and H. Shi, "A noval satellite network traffic prediction method based on GCN-GRU," in *Proc. Int. Conf. Wireless Commun. Signal Process. (WCSP)*, Oct. 2020, pp. 718–723.

[12] W. Chen, C. Zhai, X. Wang, J. Li, P. Lv, and C. Liu, "GCN- and GRU-based intelligent model for temperature prediction of local heating surfaces," *IEEE Trans. Ind. Informat.*, vol. 19, no. 4, pp. 5517–5529, Apr. 2023.

[13] X. Chen, J. Wu, and T. Wu, "The top-K QoS-aware paths discovery for source routing in SDN," *KSII Trans. Internet Inf. Syst.*, vol. 12, no. 6, pp. 2534–2553, 2018.

[14] G. Yao, Z. Dong, W. Wen, and Q. Ren, "A routing optimization strategy for wireless sensor networks based on improved genetic algorithm," *J. Appl. Sci. Eng.*, vol. 19, no. 2, pp. 221–228, 2016.

[15] C.-K. Ke, M.-Y. Wu, W.-H. Hsu, and C.-Y. Chen, "Discover the optimal IoT packets routing path of software-defined network via artificial bee colony algorithm," in *Proc. Int. Wireless Internet Conf.* TaiChung, Taiwan: Springer, 2019, pp. 147–162.

[16] B. Yong, W. Muqing, S. Jing, and Z. Min, "Optimization strategy of SDN control deployment based on simulated annealing-genetic hybrid algorithm," in *Proc. IEEE 4th Int. Conf. Comput. Commun. (ICCC)*, Dec. 2018, pp. 2238–2242.

[17] K. Truong Dinh, S. Kukliński, T. Osiński, and J. Wytrębowicz, "Heuristic traffic engineering for SDN," *J. Inf. Telecommun.*, vol. 4, no. 3, pp. 251–266, Jul. 2020.

[18] M. Shokouhifar, "FH-ACO: Fuzzy heuristic-based ant colony optimization for joint virtual network function placement and routing," *Appl. Soft Comput.*, vol. 107, Aug. 2021, Art. no. 107401.

[19] S. Sanagavarapu and S. Sridhar, "SDPredictNet—A topology based SDN neural routing framework with traffic prediction analysis," in *Proc. IEEE 11th Annu. Comput. Commun. Workshop Conf. (CCWC)*, Jan. 2021, pp. 0264–0272.

[20] W. Xiang, N. Wang, and Y. Zhou, "An energy-efficient routing algorithm for software-defined wireless sensor networks," *IEEE Sensors J.*, vol. 16, no. 20, pp. 7393–7400, Oct. 2016.

[21] T. T. Duong and L. H. Binh, "IRSML: An intelligent routing algorithm based on machine learning in software defined wireless networking," *ETRI J.*, vol. 44, no. 5, pp. 733–745, Oct. 2022.

[22] S. Mahajan, R. Harikrishnan, and K. Kotecha, "Adaptive routing in wireless mesh networks using hybrid reinforcement learning algorithm," *IEEE Access*, vol. 10, pp. 107961–107979, 2022.

[23] B. Chen, D. Zhu, Y. Wang, and P. Zhang, "An approach to combine the power of deep reinforcement learning with a graph neural network for routing optimization," *Electronics*, vol. 11, no. 3, p. 368, Jan. 2022.

[24] D. M. Casas-Velasco, O. M. C. Rendon, and N. L. S. da Fonseca, "DRSIR: A deep reinforcement learning approach for routing in software-defined networking," *IEEE Trans. Netw. Service Manag.*, vol. 19, no. 4, pp. 4807–4820, Dec. 2022.

[25] W.-X. Liu, J. Cai, Q. C. Chen, and Y. Wang, "DRL-R: Deep reinforcement learning approach for intelligent routing in software-defined data-center networks," *J. Netw. Comput. Appl.*, vol. 177, Mar. 2021, Art. no. 102865.

[26] Y. Guo, J. Chen, K. Huang, and J. Wu, "A deep reinforcement learning approach for deploying SDN switches in ISP networks from the perspective of traffic engineering," in *Proc. IEEE 23rd Int. Conf. High Perform. Switching Routing (HPSR)*, Jun. 2022, pp. 195–200.

[27] K. Zhang, X. Xu, C. Fu, X. Wang, and W. Wu, "Modeling data center networks with message passing neural network and multi-task learning," in *Neural Computing for Advanced Applications*. Guangzhou, China: Springer, 2021, pp. 96–112.

[28] P. Sun, J. Lan, J. Li, J. Zhang, Y. Hu, and Z. Guo, "A scalable deep reinforcement learning approach for traffic engineering based on link control," *IEEE Commun. Lett.*, vol. 25, no. 1, pp. 171–175, Jan. 2021.

[29] M. Tian, C. Sun, and S. Wu, "An EMD and ARMA-based network traffic prediction approach in SDN-based Internet of Vehicles," *Wireless Netw.*, vol. 2021, pp. 1–13, Aug. 2021.

[30] T. Alghamdi, K. Elgazzar, M. Bayoumi, T. Sharaf, and S. Shah, "Forecasting traffic congestion using ARIMA modeling," in *Proc. 15th Int. Wireless Commun. Mobile Comput. Conf. (IWCMC)*, Jun. 2019, pp. 1227–1232.

[31] Q. T. Tran, L. Hao, and Q. K. Trinh, "Cellular network traffic prediction using exponential smoothing methods," *J. Inf. Commun. Technol.*, vol. 18, no. 1, pp. 1–18, Jan. 2019.

[32] R. Casado-Vara, A. M. del Rey, D. Pérez-Palau, L. de-la-Fuente-Valentín, and J. M. Corchado, "Web traffic time series forecasting using LSTM neural networks with distributed asynchronous training," *Mathematics*, vol. 9, no. 4, p. 421, Feb. 2021.

[33] J. Hu, X. Wang, Y. Zhang, D. Zhang, M. Zhang, and J. Xue, "Time series prediction method based on variant LSTM recurrent neural network," *Neural Process. Lett.*, vol. 52, no. 2, pp. 1485–1500, Oct. 2020.

[34] J. Bi, X. Zhang, H. Yuan, J. Zhang, and M. Zhou, "A hybrid prediction method for realistic network traffic with temporal convolutional network and LSTM," *IEEE Trans. Autom. Sci. Eng.*, vol. 19, no. 3, pp. 1869–1879, Jul. 2022.

[35] Y. Zheke, "Prediction method of wireless network traffic based on spatiotemporal features," *J. Wireless Commun. Technol.*, vol. 31, no. 3, pp. 24–28&34, 2022.

[36] H. Shi, C. Pan, L. Yang, and X. Gu, "AGG: A novel intelligent network traffic prediction method based on joint attention and GCN-GRU," *Secur. Commun. Netw.*, vol. 2021, pp. 1–11, Sep. 2021.

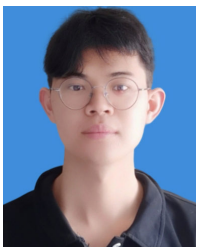
[37] Y. Li, Z.-P. Cai, and H. Xu, "LLMP: Exploiting LLDP for latency measurement in software-defined data center networks," *J. Comput. Sci. Technol.*, vol. 33, no. 2, pp. 277–285, Mar. 2018.

[38] L. Liao and V. C. M. Leung, "LLDP based link latency monitoring in software defined networks," in *Proc. 12th Int. Conf. Netw. Service Manag. (CNSM)*, Oct. 2016, pp. 330–335.

[39] X. Zhao, T. Lei, Z. Lu, X. Wen, and S. Jiang, "Introducing network situation awareness into software defined wireless networks," *KSII Trans. Internet Inf. Syst. (TIIS)*, vol. 12, no. 3, pp. 1063–1082, 2018.
[40] S. S. Bhavanasi, L. Pappone, and F. Esposito, "Routing with graph convolutional networks and multi-agent deep reinforcement learning," in *Proc. IEEE Conf. Netw. Function Virtualization Softw. Defined Netw. (NFV-SDN)*, Nov. 2022, pp. 72–77.
[41] S. Yuan-Long, L. Guang-Hong, W. Gui-Zhi, and J. Wu-Cai, "SDN traffic prediction based on graph convolutional network," *Comput. Sci.*, vol. 48, no. 6A, pp. 392–397, 2021.
[42] S.-Y. Han and T. Liang, "Reinforcement-learning-based vibration control for a vehicle semi-active suspension system via the PPO approach," *Appl. Sci.*, vol. 12, no. 6, p. 3078, Mar. 2022.
[43] S. L. Bangare, "Classification of optimal brain tissue using dynamic region growing and fuzzy min-max neural network in brain magnetic resonance images," *Neurosci. Informat.*, vol. 2, no. 3, Sep. 2022, Art. no. 100019.
[44] W. Zhang, R. Xie, Q. Wang, Y. Yang, and J. Li, "A novel approach for fraudulent reviewer detection based on weighted topic modelling and nearest neighbors with asymmetric Kullback–Leibler divergence," *Decis. Support Syst.*, vol. 157, Jun. 2022, Art. no. 113765.
[45] M. Afaq, S. Rehman, and W.-C. Song, "Large flows detection, marking, and mitigation based on sFlow standard in SDN," *J. Korea Multimedia Soc.*, vol. 18, no. 2, pp. 189–198, Feb. 2015.
[46] S. Buzura, V. Dadarlat, A. Peculea, H. Bertrand, and R. Chevalier, "Simulation framework for 6LoWPAN networks using mininet-WiFi," in *Proc. IEEE Int. Conf. Autom., Quality Test., Robot. (AQTR)*, May 2022, pp. 1–5.
[47] P. Tune and M. Roughan, "Spatiotemporal traffic matrix synthesis," in *Proc. ACM Conf. Special Interest Group Data Commun.*, Aug. 2015, pp. 579–592.
[48] *RYU*. Accessed: Mar. 20, 2023. [Online]. Available: https://ryu-sdn.org/
[49] *IPERF*. Accessed: Mar. 20, 2023. [Online]. Available: https://iperf.fr
[50] *MiniNet-WiFi*. Accessed: Mar. 20, 2023. [Online]. Available: https://mininet-wifi.github.io/
[51] *sFlow-RT*. Accessed: Mar. 20, 2023. [Online]. Available: https://sflow-rt.com/

**JINQIANG LI** was born in 1997. He is currently pursuing the master's degree with the School of Information and Communication, Guilin University of Electronic Technology. His main research interests include reinforcement learning and software-defined networking.

**MIAO YE** received the B.S. degree in theory physics from Beijing Normal University, in 2000, and the Ph.D. degree from the School of Computer Science and Technology, Xidian University, in 2006. He is currently a Full Professor and the Ph.D. Supervisor with the Guilin University of Electronic Technology. His research interests include software-defined networking, edge computing and edge storage, wireless sensor networks, and deep learning.



**LINQIANG HUANG** was born in 1998. He is currently pursuing the master's degree with the School of Computer Science and Information Security, Guilin University of Electronic Technology. His main research interests include reinforcement learning and software-defined networking.

**XIAOFANG DENG** received the B.Eng. and M.Eng. degrees in communication engineering from the Guilin University of Electronic Technology (GUET), China, in 1998 and 2005, respectively, and the Ph.D. degree from the South China University of Technology (SCUT), Guangzhou, China, in 2016. She was a Visiting Scholar with Coventry University, in 2017. She is currently an Associate Professor with the School of Information and Communication Engineering, GUET. Her research interests include cognitive networks, network economy, and information sharing.

**HONGBING QIU** received the Ph.D. degree in information and communication engineering from Xidian University, in 2004. Since 1984, he has been a Teacher of information and communication engineering with the Guilin University of Electronic Technology. From February 2012 to August 2012, he was a Visiting Researcher with the University of Minnesota, Twin Cities, USA. He is a Ph.D. Supervisor with Xidian University and the Guilin University of Electronic Technology and the Director of the Ministry of Education Key Laboratory of Cognitive Radio and Information Processing. He is a member of the Communication Theory and Signal Processing Committee of Communications Institute of China, the Broadband Wireless IP Standards Working Group, and the Home Network Standards Working Group. His major research interests include mobile communication, wireless sensor networks, and image signal processing.

**YONG WANG** was born in 1964. He received the Ph.D. degree from the East China University of Science and Technology, Shanghai, China, in 2005. He is currently a Full Professor and the Ph.D. Supervisor with the Guilin University of Electronic Technology. His main research interests include cloud computing, distributed storage systems, software-defined networks, and information security.

**QIUXIANG JIANG** was born in 1978. She received the master's degree from the Guilin University of Technology, Guilin, China, in 2005. She is a Senior Engineer with the Guilin University of Electronic Technology. Her main research interests include software-defined networks, wireless sensor networks, and cloud computing.