SDN-DRLTE Algorithm Based on DRL in Computer Network Traffic Control

Cuijie Yang, Biyan Li*
School of Automotive and Information Engineering, Guangxi Eco-Engineering Vocational & Technical College Liuzhou 545004, China
E-mail: yangcuijie1982@163.com, xxgclby@163.com
*Corresponding author

Keywords: deep reinforcement learning, software-defined network, markov decision, traffic control, off policy

Received: November 12, 2024

Affected by mobile Internet, big data and cloud computing, network traffic load is gradually increasing, and deep reinforcement learning algorithm has been widely used. To solve the uneven and congested computer network traffic, a software-defined network algorithm on the basis of deep reinforcement learning is designed, and a computer network traffic control technology is built. On the basis of traditional deep reinforcement learning algorithms, the optimal performance policy is obtained by combining Markov decision. Simultaneously, the Off Policy is introduced to establish a software-defined network traffic control model, ultimately designing a software-defined network algorithm based on deep reinforcement learning. The experimental results showed that compared with other algorithms, the designed algorithm increased the average reward value by 12.2%, 18.6%, and 6.8%, optimized by an average of 10%, and significantly increased the reward value at 3000 iterations. The average speed was higher, and the latency was significantly reduced to 6.3%. This indicated that the designed algorithm achieved the expected goals in terms of computational efficiency and network scheduling control performance. The research findings were of great significance for computer network traffic control.

Povzetek: Izboljšani algoritem SDN-DRLTE temelji na globokem ojačitvenem učenju (DRL) za nadzor prometa v računalniških omrežjih. Z uporabo Markovih odločitvenih procesov algoritem izboljšuje uravnoteženost prometa in zmanjšuje zakasnitve.

1 Introduction

Affected by Internet of Things and big data, the number and scale of network users are growing rapidly, while network services are becoming increasingly diversified and network structures are becoming more complex. Cisco's annual VNI report provides detailed forecasts and historical data on global Internet traffic. According to Cisco's 2020 VNI report, global Internet traffic is expected to grow from 122 exabytes per second in 2019 to 185 exabytes per second in 2024 [1]. In the context of explosive growth in network traffic, achieving efficient network traffic load balancing, avoiding congestion, and ensuring network service quality is crucial. Although traditional traffic scheduling and congestion solutions can control traffic to a certain extent, they are often limited by different network environments and difficult to be widely applied to complex network environment structures. In recent years, some researchers have begun to attempt to explore Deep Learning (DL) technology, providing new ideas for computer traffic control with its excellent learning and training capabilities [2]. However, the existing DL techniques have a wide range of applications and cannot provide good computational efficiency and practical application effects. Moreover, they do not consider continuous action response and cannot meet the applicability requirements in new network structures and traffic patterns [3]. Deep Reinforcement Learning (DRL) is an advanced technology in artificial intelligence, which has shown great potential in intelligent control. It cleverly combines the powerful perception ability of DL with the decision optimization ability of Reinforcement Learning (RL), providing a new way for computer network traffic control. The Software-Defined Network (SDN) has made traffic control more efficient. With the increasingly close relationship between the information society, electronics, and artificial intelligence, the complex interactions and technological advancements between these fields are becoming increasingly apparent. Gams and Kolenik explored the relationship between the laws of the information society and electronics, AI, and environmental intelligence in their research. How these technological advancements shape civilization and social structures was analyzed [4]. In this context, the research attempts to innovatively design a SDN algorithm based on DRL. Big data provides rich network usage patterns and traffic characteristics. SDN-DRLTE algorithm can use this data to optimize network traffic control strategies and achieve more accurate decisions. At the same time, SDN-DRLTE algorithm can predict the trend of network traffic, adjust the network configuration in advance, reduce congestion and improve efficiency by analyzing Internet of Things devices and big data models [5]. Combining DRL policy optimization with SDN model construction, it is aimed to optimize the control

efficiency and its practicality for different topology networks, laying a solid foundation for computer network traffic control.

2 Related work

With the growing popularity of Internet technology, network traffic is also showing a high growth trend. Traditional network traffic control methods cannot adapt to the current network environment. Therefore, some scholars have conducted relevant research on computer network traffic control methods. Aberkane et al. proposed a novel anomaly detection method based on DRL. The priority dual deep Q-network was optimized to adapt to anomaly detection problems. Video level labels were used to learn and evaluate anomalies in video clips, thereby improving detection accuracy. The experiment showed that the proposed method had higher accuracy [6]. He proposed an intelligent network traffic scheduling algorithm based on DRL and graph neural network innovation for the traffic scheduling problem in large-scale dynamic network environments. The ability of DRL in decision optimization and the advantages of graph neural networks in processing graph structured data were combined. The hierarchical RL framework enabled efficient decision-making processes from macro policies to micro-operations. The experimental results showed that the proposed algorithm significantly improved key performance indicators such as average latency, throughput, and resource utilization compared with traditional algorithms [7]. Morimoto M et al. built a neural network training model to solve practical network traffic estimation problems. The machine learning results were analyzed and the training data were expanded. Experiments showed that training neural networks could accurately estimate network traffic [8]. Dalal S et al. proposed a cloud-based Adhoc mobile model for load balancing of network traffic. The model combined alternative paths between two nodes to solve the load congestion by calculating the link and traffic load. The proposed model had a positive effect on extending the lifespan of the system [9]. Leng J et al. proposed a blockchain intelligent autonomous process control method for network personalized control problems. Through blockchain intelligent pyramid and a series of decentralized control modes, personalized demand scheduling could be achieved. The experiment showed that the proposed method had good applicability for flow control [10].

Computer network traffic control methods often incorporate artificial intelligence technology for research. Some scholars have conducted research on DRL and SDN algorithms. Mahmood T et al. built a smart fault detection routing technology based on RL to develop energy-saving routing protocols for learning in wireless networks. This method overcame the energy loss of transmitting data by reducing the remaining energy of cluster nodes in the network. The method could effectively improve the robustness of the network [11]. Kosanoglu F et al. built a combination algorithm relying on DRL and simulated annealing to improve asset reliability and reduce maintenance costs. The optimal neighbor structure was selected by transmitting the optimal solution to the simulated annealing algorithm as the initial solution. The results indicated that the proposed algorithm had superiority in finding solutions [12]. Li H proposed a secure DRL method to optimize the operation of distribution networks. The Markov decision process was formalized and constrained. A constraint strategy was adopted to optimize the training network and achieve cost minimization. The results showed that DRL methods had good stability [13]. Bhardwaj S proposed the SDN approach to improve network resource utilization. This strategy improved the efficient traffic routing by controlling open-source controllers. The method could achieve good network performance [14]. Zhang D et al. designed a dynamic task offloading approach on the ground of DRL to achieve offloading computation with low task latency and low energy consumption. On the basis of improving traditional Q-learning algorithms, DL and RL were combined. This algorithm had better performance in energy consumption [15]. The summary and analysis of existing research methods are shown in Table 1.

Table 1: Summary and analysis of existing research methods.

Reference	Method	Main Results	Limitations	Advantages of SDN-DRLTE
Aberkane et al. [6]	Priority Duel Deep Q Network Anomaly Detection	Accurate detection	Video level labels are required	No need for a large amount of annotated data
He [7]	DRL and GNN traffic scheduling	Improve latency, throughput, and utilization efficiency	Environmental adaptability to be verified	Excellent robustness
Morimoto M et al. [8]	Neural network traffic estimation	Accurate estimation	Data dependency	Automatic strategy optimization
Dalal S et al. [9]	Cloud Adhoc load balancing	Resolve congestion	Fixed path	Dynamic traffic adjustment
Leng J et al. [10]	Blockchain intelligent control	Personalized demand scheduling	Distributed control is complex	Simplify control processes and improve applicability

Mahmood T et al. [11]	RL fault detection routing	Energy saving routing	Wireless network limitations	Widely applicable in various environments
Kosanoglu F et al. [12]	DRL+simulated annealing	Asset optimization	Computationally intensive	Improve computational efficiency
Li H [13]	Safe DRL distribution network	Cost minimization	Distribution network limit	Wide flow control
Bhardwaj S [14]	SDN resource utilization	SDN resource utilization	Lack of performance analysis	In depth analysis of performance
Zhang D et al. [15]	DRL task uninstallation	Low latency energy consumption	Internet of Things Limitations	Widely applicable to the internet

In summary, although some scholars have conducted relevant research on computer network traffic control, the powerful ability of DRL in decision optimization and the flexibility of SDN in network traffic management have not been fully utilized. Therefore, a SDN algorithm based on DRL is proposed to automatically learn and optimize decision strategies, adapt to dynamic changes in network traffic, and demonstrate performance beyond existing technologies in practical applications to provide assistance in improving computer network performance.

3 SDN-DRLTE algorithm based on DRL in computer network traffic control

3.1. Traffic control model architecture based on DRL and SDN

Faced with the rapidly growing traffic in computer networks, achieving network load balance through reasonable control and scheduling has become an increasingly vital research topic in computer networks. Due to the constantly changing network structures and traffic patterns, traditional traffic scheduling solutions are becoming increasingly inadequate. Therefore, a widely applicable intelligent network traffic control scheme is necessary [16]. In recent years, DRL algorithm has made breakthrough progress in multiple fields, especially in computer science. It combines the feature representation capability of DL with the decision-making capability of RL to form a powerful machine learning paradigm. DRL can optimize decision strategies through automatic exploration and learning by agents in the environment without making clear rules and guidance. It also indicates that DRL can provide opportunities for achieving more complex network traffic control. The DRL algorithm process is shown in Figure 1.

In Figure 1, the process first sets up the environment, defines the environment in which the agent is located such as the state space reward function. Then, the deep neural network is initialized. The intelligent agent retrieves states from the environment. After selecting and executing actions on the basis of the state, the environment transitions to a new state and corresponding reward values are given. Finally, the state, actions, and reward data of the interaction process are recorded to form a training dataset. This process is repeated until the training is completed. The study sets the environment as a finite Markov Decision Process (MDP) decision. When the policy rules are uncertain, DRL selects the optimal performance policy from them. The optimal performance policy is shown in equation (1).

$$\pi^* = \arg\max J(\pi) \tag{1}$$

In equation (1), π^* signifies the performance of the maximum policy. π signifies action policy. $J(\pi)$ represents the expected return of the policy. The updated policy parameter is shown in equation (2).

$$\theta_{k+1} = \theta_k + \alpha \nabla_\theta J(\pi_\theta)\Big|_{\theta_k} \tag{2}$$

In equation (2), θ_{k+1} represents the updated policy parameter. θ_k signifies the parameter of the original policy. $\nabla_\theta J(\pi_\theta)$ signifies the gradient of policy performance. π_θ represents the form of action policy. α represents a constant. To apply the algorithm in practical, it is necessary to transform the gradient representation of the policy. The policy performance gradient is displayed in equation (3).

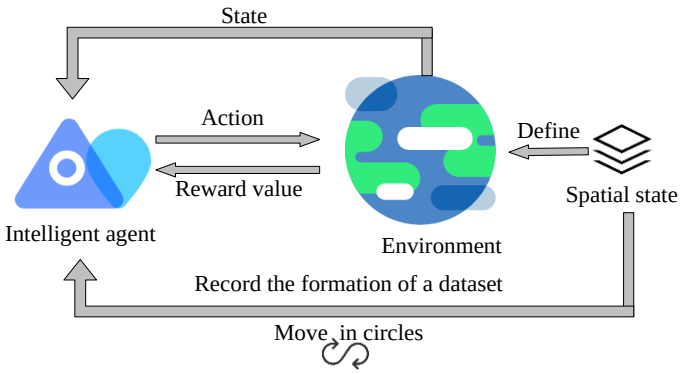


Figure 1: DRL algorithm flowchart.

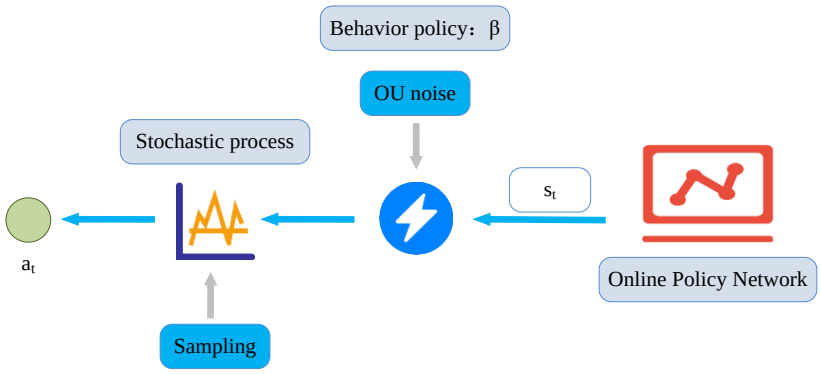


Figure 2: Exploration of random actions in off policy.

$$P(\tau|\theta)=\rho_0(s_0)\Pi_{t=0}^{T}P(s_{t+1}|s_t,a_t)\pi_\theta(a_t|s_t) \tag{3}$$

In equation (3), $\tau=(s_0,a_0,\ldots,s_{T+1})$ represents the sequence of state actions obtained by executing the policy once. T represents the time step of operation. t signifies the times that the policy is executed. ρ_0 represents the initial state distribution. s represents the state. a represents the action. $P(\tau|\theta)$ represents the probability of executing policy π_θ to generate τ. At this time, the transition state of MDP needs to be independent of the previous state and only linked with the current state action. The gradient of parameter $P(\tau|\theta)$ with respect to θ is displayed in equation (4).

$$\nabla_\theta P\left(\tau|\theta\right)=P\left(\tau|\theta\right)\nabla_\theta\log P\left(\tau|\theta\right) \tag{4}$$

In equation (4), $\nabla_\theta P\left(\tau|\theta\right)$ signifies the gradient of the probability of generating the execution policy with respect to the parameter. The logarithm of equation (3) obtains (5).

$$\nabla_\theta\log P\left(\tau|\theta\right)=\sum_{t=0}^{T}\nabla_\theta\log\pi_\theta(a_t|s_t) \tag{5}$$

After simplification, equation (6) is obtained.

$$\nabla_\theta J(\pi_\theta)=\underset{\pi}{E}\left[\sum_{t=0}^{T}\nabla_\theta\log\pi_\theta(a_t|s_t)R(\tau)\right] \tag{6}$$

In equation (6), $R(\tau)$ represents the state action sequence obtained by executing the policy. The gradient of policy performance is an expectation, indicating that the sample mean value is used for estimation calculation. The sample mean of the strategy performance gradient is shown in equation (7).

$$\hat{g}=\frac{1}{|D|}\sum_{\tau\in D}\sum_{t=0}^{T}\nabla_\theta\log\pi_\theta(a_t|s_t)R(\tau) \tag{7}$$

In equation (7), $\hat{g}$ represents the estimated value of the policy performance gradient. $|D|$ represents the size of the collected dataset D. An offline policy training method called Off Policy random action exploration is adopted, as shown in Figure 2.

In Figure 2, a_t represents exploration data. s_t represents online data. β represents strategic behavior. Off policy training obtains random processes through data sampling as introduced random noise, and then uses a policy network for training to obtain the optimal policy in the dataset and improve the temporal correlation of each data. Meanwhile, SDN has made network traffic control more efficient and convenient [17].

SDN is a new network architecture, which enhances the network flexibility and manageability by separating the control plane from the data forwarding plane and introducing programmability. The SDN is displayed in Figure 3.

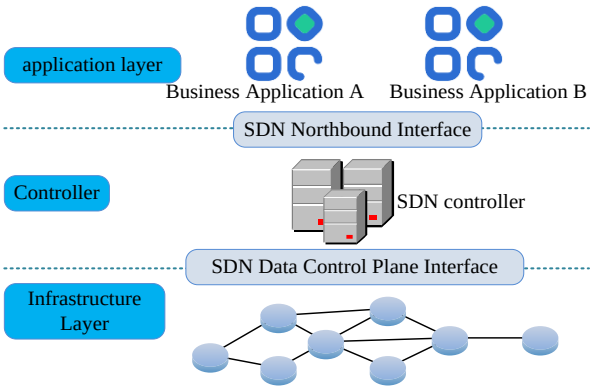


Figure 3: SDN architecture diagram.

In Figure 3, the SDN has three layers: infrastructure layer, control layer, and application layer. The complex application layer communicates with business applications through the SDN northern interface. Management personnel can dynamically adjust network behavior through flat interfaces, convert application layer commands into rules, and send them to devices. The control layer, namely the control plane, undertakes centralized management of decision-making processes such as policies, routing, and traffic control in the network. The infrastructure layer, that is, the data forwarding plane, undertakes actual packet forwarding and processing. This design allows forwarding devices to focus more on high-speed and low latency data transmission, while delegating complex control logic to the control plane for processing. The SDN architecture provides flexibility and programmability for network traffic control by separating the control plane and data plane. In control plane integration, SDN controllers act as the brain of intelligent agents, making decisions and dynamically adjusting network traffic based on network status. Combining the learning mechanism of DRL with the programmability and dynamic suitability of SDN, it is possible to quickly adjust traffic control strategies when dealing with large and complex network traffic.

3.2. Design of SDN-DRLTE intelligent traffic control based on DRL

When conducting computer network traffic control, continuous action reactions are required. Deep Deterministic Policy Gradient (DDPG) is a policy-based DRL algorithm widely used to solve decision problems in continuous action spaces [18]. Therefore, the study uses the DDPG to solve continuous control problems in computer networks. In the DDPG, the experience replay mechanism of deep Q-networks is borrowed, which can significantly enhance the stability and learning efficiency. Meanwhile, the experience replay mechanism stores the experience samples generated by the agent during the exploration of the environment. These samples are randomly sampled during the training to update the network, achieving efficient sampling and utilization of samples, which can effectively improve the learning performance and stability. The DDPG algorithm is displayed in Figure 4.

In Figure 4, the DDPG algorithm consists of two networks, the Actor network and the Critic network. The Actor network is responsible for selecting actions. The Critic network evaluates the value of these actions. The algorithm stores the experience of the intelligent agent in the replay buffer for training the network, which helps to improve sample efficiency and break down correlations between samples. For stable training, the DDPG algorithm obtains data from the SDN network environment and inputs it into the Actor network. Then, the output data is stored in the replay buffer as experience for sampling. The Actor network inputs data into the Critic network by selecting actions, and the data is processed and returned. In practical applications, DDPG typically explores the environment by adding noise to discover better strategies. The Actor network consists of an input layer, two hidden layers, and an output layer. The hidden layer has 128 and 64 neurons in each layer, with ReLU as the activation function, and Tanh as the activation function for the output layer. The Critic network accepts a combination of states and actions, consisting of two hidden layers and an output layer without an activation function. Important hyperparameters include learning rate, where Actor is 0.001, Critic is 0.002, discount factor is 0.9~0.9, experience replay buffer size is (1,000,000), and batch size is 32~128. The update frequency of the target network is 100 iterations. These settings ensure that the model can effectively learn and optimize traffic control strategies in dynamic network environments. The parameter of the policy network is shown in equation (8).

$$\nabla_\theta J = E\left[\nabla_a Q(s,a)\Big|_{s=s_t,a=\mu(s_t)} \nabla_\theta \mu(s)\Big|_{s=s_t}\right] \quad (8)$$

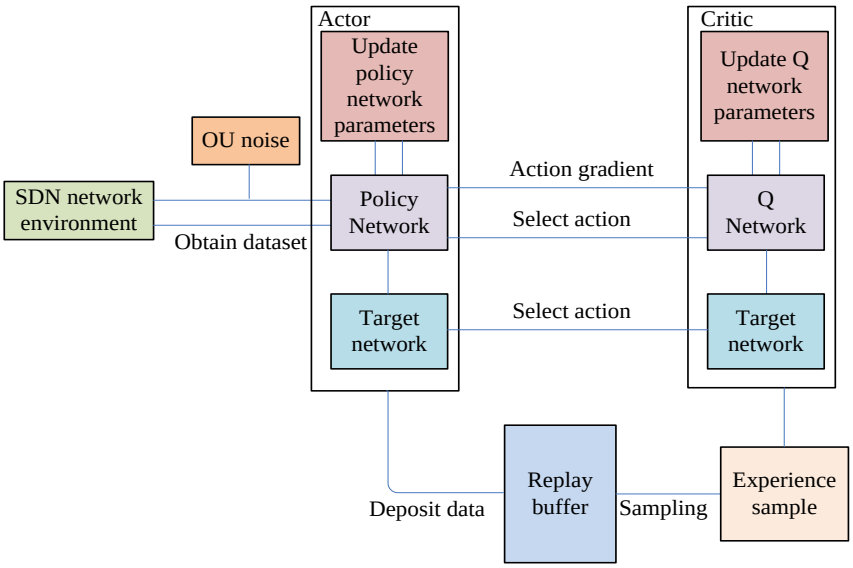


Figure 4: DDPG algorithm structure diagram.

In equation (8), $\nabla_\theta J$ represents the differentiation between the expected cumulative reward J and the parameter θ in the policy network. $Q(s,a)$ represents the evaluation network. $\mu(s)$ represents the policy network. s_t signifies the state at moment t. $\mu(s_t)$ signifies the action taken at t. The evaluation network is updated using the time difference method, which updates and optimizes relevant parameters by comparing the differences between two-time steps or states before and after, in order to approximate the optimal policy. Traditional DDPG algorithms often rely on simple random noise during the exploration phase to ensure sufficient exploration. However, in complex flow control environments, this random exploration is inefficient and hard to quickly adapt to dynamic changes in the environment [19-20]. Therefore, the study introduces an exploration policy based on Traffic Engineering (TE) perception, which utilizes real-time state information of the traffic environment to dynamically adjust the exploration intensity or direction. At the same time, in order to solve the neglected important experience and single step benefits, a priority-based experience replay pool is adopted and a multi-step benefit prediction mechanism is introduced to jointly improve the applicability and training efficiency of the DDPG algorithm in traffic control. Standard DDPG mainly focuses on single step returns, which may lead to algorithms overly focusing on short-term benefits and neglecting long-term benefits. The multi-step benefit prediction mechanism predicts long-term benefits by considering the cumulative returns of multiple future time steps, enabling the algorithm to better balance short-term and long-term goals. When facing complex and dynamic network environments. This mechanism enhances the foresight and adaptability of the algorithm, enabling it to more effectively handle long-term optimization problems of network traffic. The differential error of event experience is prioritized, as defined in equation (9).

$$p_i = y - Q(s_t, a_t) \tag{9}$$

In equation (9), p_i represents the priority of experience i. $Q(s_t, a_t)$ represents the evaluation network at a certain moment. y represents the target value for evaluating the network, as shown in equation (10).

$$y = r + \gamma Q^*(s_{t+1}, \mu^*(s_{t+1})) \tag{10}$$

In equation (10), r signifies the reward function. γ represents the discount coefficient. The probability of obtaining experience from the experience pool is shown in equation (11).

$$P(i) = \frac{p_i^{\alpha_0}}{\sum_k p_k^{\alpha_0}} \tag{11}$$

In equation (11), $P(i)$ signifies the probability of obtaining experience i from the experience pool. α_0 represents the degree to which control priority is used for probability calculation. k represents a constant. When the control priority is not used for calculation, the algorithm has a unified sampling operation. The multi-step return is equation (12).

$$R_t^{(n)} = \sum_{k=0}^{n-1} \gamma_t^{(k)} R_{t+k+1} \tag{12}$$

In equation (12), $R_t^{(n)}$ represents multi-step returns. n represents the number of steps for multi-step returns. After achieving the effect of accelerating reward feedback, the Critic network is updated. First, the network target value is shown in equation (13).

$$y_t = R_t^{(n)} + \gamma^{(n)} Q(s_{t+n}, \mu^*(s_{t+n})) \tag{13}$$

In equation (13), y_t represents the final evaluation network objective. In the DDPG algorithm, the design of the reward function needs to reflect the key performance

indicators of network traffic control, while considering fairness and priority processing. First, it is necessary to ensure fair resource allocation and traffic balance.

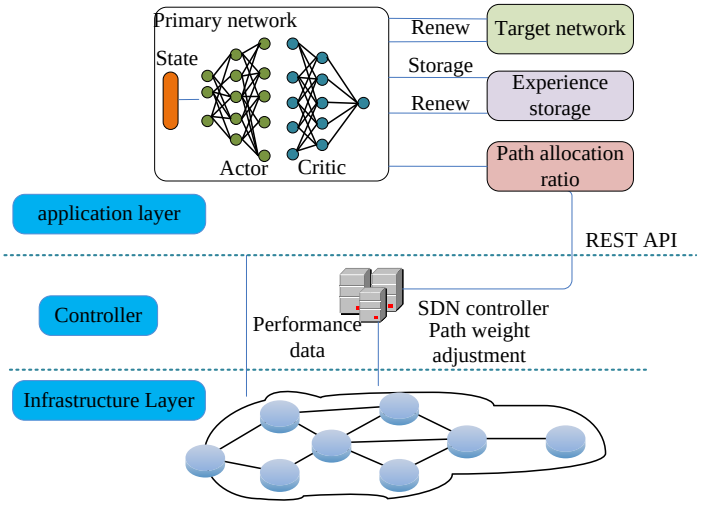


Figure 5: SND flow control model based on DRL.

SDN-DRLTE

C1: actor networklearning rate n", critic network learning rate n,target network learning rate t, mini-batch size b, replay size N, episode step M, time step T.
C2: Initialize the parameter actor network and critic network respectively.
C3: Initialize prioritized replay buffer R with size N.
C4: Initialize a random process N for action exploration.
C5: for episode =1 to M do
C6: if size of R≥b then
C7: Update the parameter of critic network
C8: end if
C9: end for
C10: Update the parameter of target network
C11: end for

Figure 6: SDN-DRLTE traffic control algorithm.

Therefore, all users or services can receive reasonable bandwidth allocation and avoid unfair treatment of certain users or services due to resource competition. A reward function is used to encourage balanced traffic distribution in the network. Some links can be overloaded while others are idle, thereby improving overall network efficiency. In terms of priority processing, for applications that require high service quality, the reward function can provide higher rewards for traffic allocation that meets specific requirements. When dealing with emergency traffic in the network, the reward function can provide additional positive rewards for prioritizing the processing of these traffic. Meanwhile, the reward function can dynamically adjust weights based on network status and external conditions to adapt to constantly changing network demands and business priorities. The SND flow control model based on DRL is shown in Figure 5.

In Figure 5, the flow control model is based on the architecture of these two algorithms. The decision-making layer interacts with other layers using the DRL flow control algorithm. The performance data information output from the forwarding layer is input at different times, processed by the policy network to output actions, and then transmitted to the SDN controller. The control layer mainly accepts actions issued by the decision-making layer, adjusts path weights, and inputs them to the forwarding layer. The forwarding layer is responsible for transmitting the performance data of each business flow to the decision layer, completing the SND traffic control architecture based on DRL. The designed model selects the global optimal policy for computing the network through DRL algorithm. The SDN controller is used to provide corresponding network traffic information for traffic scheduling decisions, achieving traffic scheduling and control. The SDN-DRLTE traffic control algorithm is shown in Figure 6.

In Figure 6, at the beginning of each training round, an initial state is obtained. After obtaining variables for

input, the main network is copied to obtain a target network. Then, the probability selection is performed at each time step. After receiving rewards and status, multi-step returns are calculated and the experience is finally stored in the experience pool. By carefully selecting model parameters, the SDN-DRLTE algorithm demonstrates better learning performance and faster convergence speed. Learning rate is a key parameter in DRL algorithm, which affects the convergence speed and stability of the algorithm. A moderate learning rate is selected to ensure that the algorithm can strike a balance between exploration and utilization. A high learning rate may lead to algorithm instability, while a low learning rate may result in slow convergence speed. The selection of network architecture is based on the characteristics of the DDPG algorithm, which needs to handle continuous action spaces. The study introduces a structure that includes an Actor Network and a Critic Network, each of which contains multiple layers of neural networks. This architecture can capture complex network state characteristics and provide effective decision-making for network traffic control. The exploration strategy is crucial for DRL algorithms, as it determines how the algorithm explores in unknown environments. The study incorporates an exploration strategy based on traffic engineering perception, which can dynamically adjust the exploration intensity according to real-time network status to improve the adaptability and efficiency. This strategy is selected based on the practical needs of network traffic control to ensure that the algorithm can remain effective in the constantly changing network environment. Experience replay is a commonly used technique in DRL algorithms to improve the sample utilization efficiency. The study selects a priority experience replay pool, which can adjust the sampling probability based on the importance of experience, thereby improving learning efficiency. When deploying the SDN-DRLTE algorithm in practical systems, DL and RL techniques may require high computational resources during training and inference processes. In real-time network traffic control, high computational overhead may affect the response speed and scalability of algorithms. The research aims to improve the algorithm efficiency through hardware accelerators such as GPUs or TPUs to handle computationally intensive tasks. Then, distributed computing is used to distribute computing tasks to multiple servers or edge computing is used to reduce the burden on the central node. The SDN-DRLTE algorithm obtained can achieve parallel utilization of resources while controlling network traffic policies, and improve the learning speed.

4 Validity analysis of SDN-DRLTE algorithm in computer network traffic control

4.1 Performance analysis experiment of computer network traffic control based on SDN-DRLTE

To analyze the effectiveness of the designed SDN-DRLTE, the focus is on the convergence rate and latency effect of the algorithm. The study forms a policy network consisting of two fully connected layers, with 28 and 14 neurons in each layer. Simultaneously, a fully connected layer is used to process the state of the evaluation network and a fully connected layer is taken to process the actions of the evaluation network, with 14 neurons set. The experimental environment is shown in Table 2.

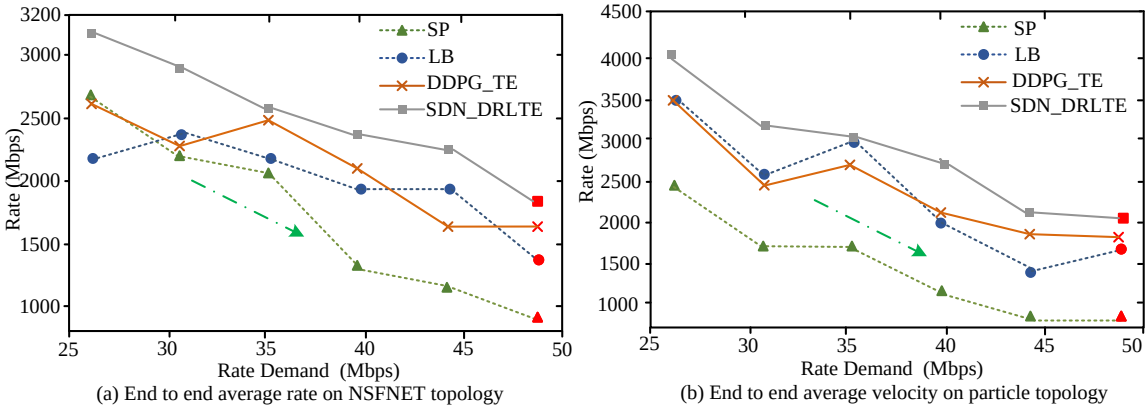
In the hardware configuration of the experiment, the service is an Intel Xeon Gold 6148 processor, 2.4 GHz, 20 cores, 256 GB DDR4 memory, 4 TB NVMe SSD storage, and 100 Gbps network interface card. In the software configuration, the operating system is Ubuntu 20.04 LTS, the SDN controller is Ryu SDN Framework, the network simulator is Mininet 3.0.0, the DL framework is TensorFlow 2.0, and the programming language is Python 3.8. The important parameters set in the study are N=1200, b=52, n=4, and T=12. α=0.6. This value usually strikes a balance between exploration and utilization, neither converging too quickly to suboptimal strategies nor being too conservative. β=0.5. This value is selected based on experimental results and the characteristics of network traffic control problems, aiming to balance short-term and long-term rewards. The model is simulated on two network topologies, namely the Network Simulation Framework for ET (NSFNET) and the Particle Experiment Network Topology. 18 OllyDbg (OD) pairs for selecting the shortest path in each topology, and 100 different streams for computer ports are set. The initial traffic window is [15, 35]Mbps. Three commonly used methods are compared with the SDN-DRLTE traffic control algorithm based on DRL, as shown in Table 3.

Table 2: Computer network traffic control experimental environment.

	Decision-making	Controller	Forwarding layer
Host configuration	Memory: 300GB Hard drive: 15T Cores: 8	Memory: 300GB Hard drive: 15T Cores: 8	Memory: 300GB Hard drive: 15T Number of cores: 8
Simulation	Select	Ryu	Mininet
Number of neurons	N/A	42	14
Activation function	/	/	sigmod

Table 3: Experimental comparison method.

Experimental methods	Algorithm description	Advantage	Disadvantages
Shortest Path (SP)	Each OD pair uses the shortest path to transmit traffic	The algorithm is simple, the computational cost is low, and it performs well in static network environments.	Not considering link load may result in some links being overloaded while others are idle.
Load Balance (LB)	Each OD evenly transmits traffic on a transportable path	Effectively avoiding single link overload and improving overall network throughput. It performs well in multipath environments.	For rapidly changing network environments, dynamic adjustments may have delays and responses may not be timely enough.
DDPG Based on TE (DDPG-TE)	Each OD is generated using the original DDPG algorithm The TE scheme controls traffic. The parameters and network structure are the same as SDN-DRLTE	Capable of handling complex network environments, with strong adaptability and the ability to adjust strategies in real-time.	A large amount of computing resources are required, and there may be delays in environments with high real-time response requirements.



(a) End to end average rate on NSFNET topology

(b) End to end average velocity on particle topology

Figure 7: Average end-to-end rate on different topologies.

The reason for choosing SP as the baseline is that it represents the most fundamental method in network routing. It is easy to implement, and has low computational costs. The network topology node is set to 14, with a link capacity between 100 Mbps and 1 Gbps, and a traffic demand between 10 Mbps and 100 Mbps. LB is chosen as the baseline because it represents a simple traffic allocation strategy that reduces network congestion by evenly distributing traffic. The network topology is the same as SP, with a fixed traffic requirement of 50 Mbps for each node pair. DDPG-TE combines deep deterministic policy gradients and traffic engineering, providing a RL solution similar to SDN-DRLTE but not fully optimized. The network topology remains the same, with traffic requirements ranging from 10 Mbps to 200 Mbps. When evaluating the performance of various methods, two key performance indicators are used: the sum of the rate and latency of all OD pairs within a single time step, and the sum of the accumulated rewards for all time steps throughout the entire turn. After training the policy network, the experimental results are obtained by running it for 10 rounds and calculating the average. This helps to visually display the applicability and performance of various methods in different network environments, ensuring the stability and reliability of the evaluation.

The end-to-end average rates of the four algorithms obtained from the experiment on NSFNET and particle experiment network topology are shown in Figure 7.

In Figure 7, when the demand for each OD rate increased, the overall average transmission rate showed a downward trend. From Figure 7 (a), when using the NSFNET topology, the SP algorithm had the fastest rate drop, followed by the LB, DDPG-TE, and SDN-DRLTE algorithms, all of which had the lowest average rate drop when the rate requirement reached 50 Mbps. The SDN-DRLTE algorithm has a higher average network rate compared with other algorithms. It is because the SP algorithm only considers the shortest path and ignores the current load condition of the link, which may cause some links to overload under high load, thereby significantly reducing the overall speed. From Figure 7 (b), the performance of each algorithm on the particle experiment network topology is roughly the same as on the NSFNET topology, but with a higher starting rate. The results indicate that the rate reduction of SDN-DRLTE is significantly smaller than other methods. This is because SDN-DRLTE can dynamically adjust traffic

allocation strategies based on real-time network environments, thereby reducing performance degradation caused by resource bottlenecks. In actual data center networks, faced with high-density traffic and complex traffic patterns, SDN-DRLTE algorithm can adjust traffic allocation in real-time, optimize network resource utilization, reduce congestion points within data centers, and improve overall network efficiency. In the context of the Internet of Things, SDN-DRLTE improves the utilization efficiency of network resources, reduces the need for additional hardware, and lowers deployment and operational costs. The end-to-end average latency of the four algorithms on NSFNET and particle experiment network topology is shown in Figure 8.

In Figure 8, as the demand for network speed increased, the overall latency also increased. From Figure 8 (a), under the NSFNET topology, the overall latency increase of LB was the highest, the average latency difference between LB and DDPG-TE was not significant. The latency increase of SDN-DRLTE was the least, only at 6.3%. The reason is that the LB algorithm may not be able to respond quickly to changes in link state when facing network attacks, resulting in delayed traffic reallocation and increased latency. From Figure 8 (b), the overall performance trend on the topology of the particle experimental network was also roughly the same. Compared with other algorithms, SDN-DRLTE significantly reduced the average end-to-end latency. This indicates that SDN-DRLTE effectively alleviates network congestion and reduces packet waiting time through its intelligent decision-making and traffic optimization mechanism, thereby achieving excellent performance with low latency even under high rate requirements. In data center environments, low latency is crucial for maintaining service responsiveness and reliability. SDN-DRLTE algorithm reduces network latency through intelligent decision-making, which is particularly important for applications that require fast data transmission. In the context of the Internet of Things, data security and transmission reliability are crucial.

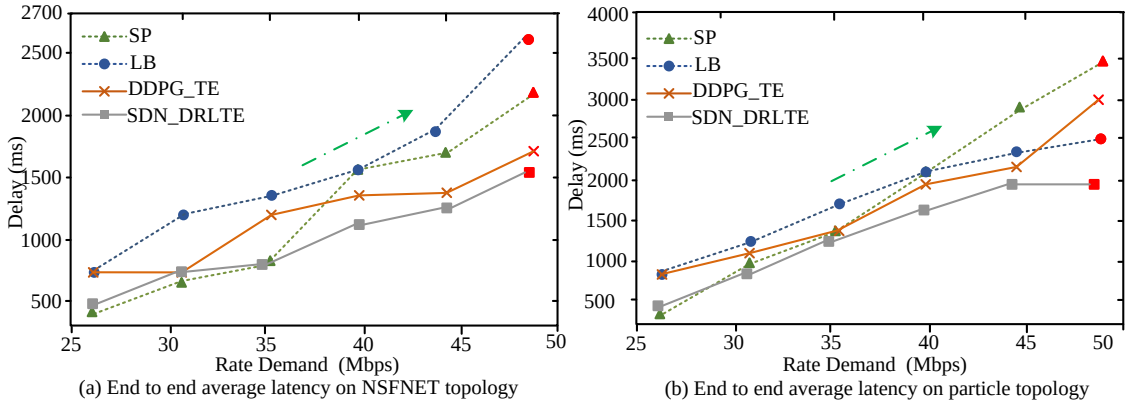


Figure 8: Average end-to-end latency on topology.

SDN-DRLTE algorithm optimizes traffic control latency, reduces potential attack surfaces, and ensures the stability and reliability of critical communication. Then, the algorithm performance is tested under different traffic loads of low, medium, and high to evaluate its adaptability and efficiency under different load conditions. The performance of SDN-DRLTE algorithm under different traffic loads is shown in Table 4.

According to Table 4, SDN-DRLTE algorithm exhibited excellent performance and stability under different traffic load conditions. Under low load conditions, the network achieved the highest average throughput and lowest latency, with extremely low packet loss rate and moderate network utilization. This demonstrates that the algorithm can effectively utilize network resources without causing overload. With the increase of traffic load, although the throughput slightly decreases and the latency and packet loss rate increase, SDN-DRLTE algorithm can still maintain high network utilization and excellent algorithm adaptability. This indicates that the algorithm can adapt to different load conditions, maintain efficient network operation, and maintain good performance even under high load conditions. Then, the network simulation tool NS-3 is used to create a dynamically changing network environment. The performance evaluation of the DN-DRLTE algorithm under dynamic network changes is shown in Table 5.

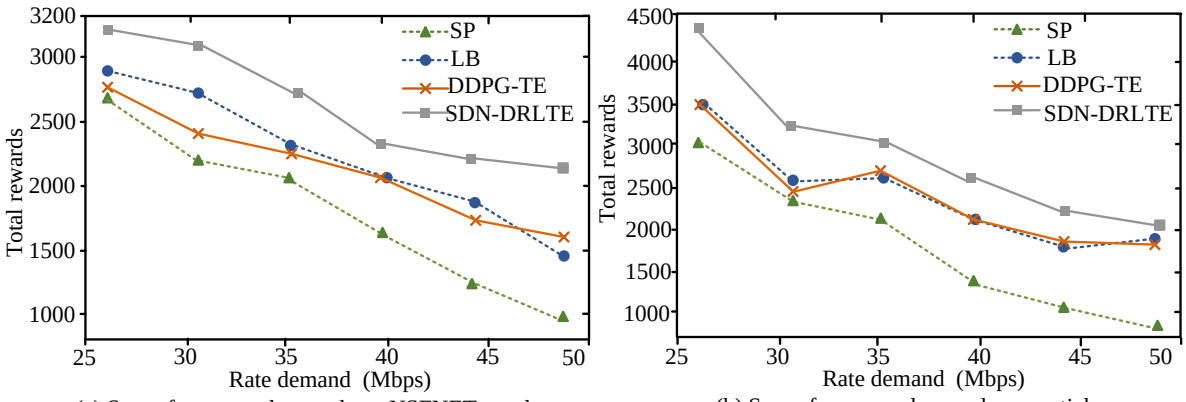
In Table 5 the SDN-DRLTE algorithm exhibited good robustness in dynamic network changes. Whether it is link failures, sudden traffic, changes in user behavior, or network attack simulations, algorithms can quickly adapt to these changes, maintaining high throughput and low latency. Although the packet loss rate has increased in some cases, the network utilization rate still remains at a high level. This indicates that the algorithm can effectively manage and allocate network resources to cope with emergencies. The rapid recovery of adaptation time demonstrates the algorithm's ability to respond quickly in emergency situations such as network attacks and link failures. Overall, the performance of SDN-DRLTE algorithm under dynamic network changes demonstrates its practicality and reliability in practical network environments, ensuring stable and efficient operation of the network in various unexpected events.

Table 4: Performance under different traffic loads.

Traffic load	Average throughput (Mbps)	Average latency (ms)	Packet loss rate (%)	Network utilization rate (%)	Algorithm adaptability
Low load	950	2	0.1	50	Excellent
Medium load	890	5	0.3	60	Excellent
High load	780	10	0.5	75	Excellent

Table 5: Performance under dynamic network changes.

Network dynamics	Average throughput (Mbps)	Average latency (ms)	Packet loss rate (%)	Network utilization rate (%)	Adaptation time (s)	Algorithm robustness
Link failure	950	5	0.20	65	10	Excellent
Sudden traffic	980	4	0.10	70	8	Excellent
Changes in user behavior	960	3	0.15	68	12	Excellent
Network attack simulation	920	6	0.25	60	15	Excellent



(a) Sum of one round rewards on NSFNET topology

(b) Sum of one round rewards on particle experiment network topology

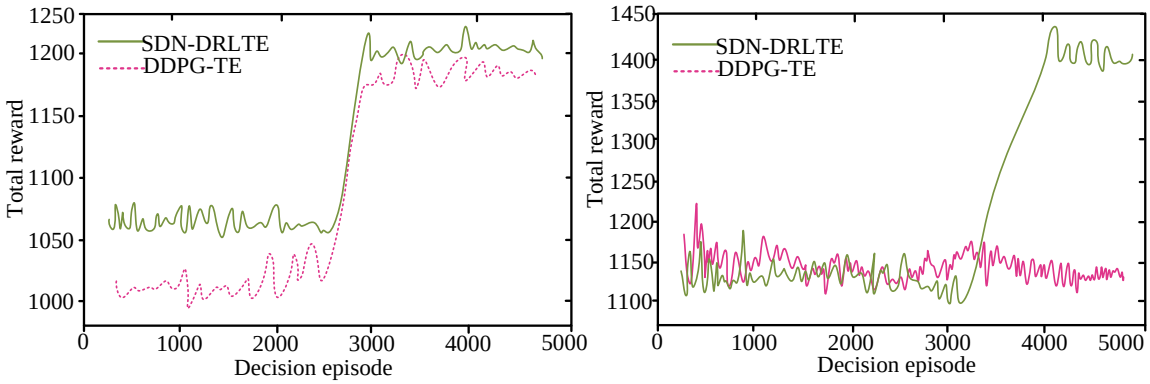
Figure 9: Sum of rewards for one round on topology.

4.2 Actual improvement results of traffic control based on SDN-DRLTE

After analyzing the effectiveness of the SDN-DRLTE algorithm, to further verify its performance and advantages in practical applications, a series of experimental tests and improvement result analysis are conducted. This process is crucial to ensure that the algorithm can achieve the expected results in actual deployments. The experiment collects various performance indicators data during the experimental process using network analysis tools, log systems, and specialized testing tools. The reward results and algorithm improvement results of each algorithm are analyzed. Firstly, Confidence interval estimation is performed. Under the same conditions, each algorithm is subjected to multiple experiments to collect sufficient data points. Then, their average reward value and standard deviation are calculated. A normal distribution is used to construct a confidence interval for the average reward value. The sum of one round rewards for four algorithms on NSFNET and particle experiment network topology is shown in Figure 9.

From Figure 9, in the reward values of each round, all algorithms decreased as the rate increased. From Figure 9 (a), under the NSFNET topology, the reward values of all algorithms showed a downward trend, with SP decreasing the fastest. SDN-DRLTE was more stable compared with other algorithms. From Figure 9 (b), the downward trend on the particle experimental network topology was similar to that under the NSFNET topology. Each algorithm had a faster descent rate and a higher initial reward value. The average reward value of SDN-DRLTE was 12.2%, 18.6%, and 6.8% better than LB, SP, and DDPG-TE. The confidence interval was [6.5%, 18.9%]. At a 95% confidence level, the p-value of the t-test was 0.03. From the reward results, with the dynamic changes in traffic demand, the adjustment of SDN-DRLTE appeared relatively slow. SDN-DRL could actively learn and adapt to changes in the network environment, making decisions through its built-in policy

network. This process not only involves a deep understanding of the current network state, but also involves predicting future traffic trends and generating corresponding optimization strategies. The training process of DRL algorithm on NSFNET and particle experiment network topology is shown in Figure 10.



(a) Training process of DRL algorithm on NSFNET topology

(b) Training process of DRL algorithm on particle network topology

Figure 10: The training process of DRL algorithm on topology.

From Figure 10, the SDN-DRLTE algorithm had a faster convergence speed and higher reward value compared with the DDPG-TE. According to Figure 10 (a), as the number of iterations increased, the reward values of both algorithms showed a relatively stable fluctuation state within 3000 iterations. The fluctuation of DDPG-TE algorithm may be a natural phenomenon that occurs when the algorithm seeks the best balance between exploration and utilization. The fluctuation of SDN-DRLTE algorithm may reflect the trade-off between algorithm exploration and utilization. In the early stages of training, algorithms may focus more on exploration to understand the environment and find better strategies. However, when the number of iterations reached 3000, there was a significant change in the reward values. SDN-DRLTE algorithm showed a sudden increase in reward values. This trend continued to persist in the subsequent iteration process, ensuring that the reward value of the SDN-DRLTE algorithm remained at a higher level. In Figure 10 (b), regardless of how the iteration increases, the reward value of the DDPG-TE exhibited a relatively stable fluctuation, with roughly the same upward and downward trends. The reward value of the SDN-DRLTE algorithm suddenly increased when the number of iterations reached around 4,000. With the flexibility of SDN and the intelligence of DRL, SDN-DRLTE algorithm can more effectively cope with complex and changing network environments, achieving efficient and intelligent traffic engineering optimization. With the flexibility of SDN and the intelligence of DRL, the adaptive characteristics of SDN-DRLTE algorithm can more effectively cope with complex and changing network environments. Especially in data center networks and IoT environments, SDN-DRLTE algorithm can easily adapt to changes in network size and environment, and automatically adjust traffic control strategies to maintain network performance.

5 Discussion

The study aims to establish a SDN traffic control model and design an SDN-DRLTE algorithm based on DRL. Compared with the DRL combined simulated annealing algorithm proposed by Kosanoglu F et al [12], this algorithm can further improve computational efficiency. The reason is that the proposed algorithm focuses on network traffic control, which can improve network performance and resource utilization. Compared with SDN resource utilization method proposed by Bhardwaj S [14], the model proposed in this paper can deeply analyze performance, reduce network latency, and avoid congestion. The reason is that the model in this article achieves intelligent control and optimization of network traffic through DRL technology. The end-to-end experiments show that as the demand for speed increases among various ODs, the average network speed of the SDN-DRLTE algorithm is higher compared with other algorithms. SDN-DRLTE can dynamically adjust traffic allocation strategies based on real-time network environments, thus reducing performance degradation caused by resource bottlenecks. In the delay experiment, with the increase of network speed requirements, the delay of SDN-DRLTE increased the least, only at 6.3%, effectively alleviating network congestion. The average reward value of SDN-DRLTE was 12.2%, 18.6%, and 6.8% better than LB, SP, and DDPG-TE algorithms. SDN-DRLTE can actively learn and adapt to changes in the network environment. In the comparison of reward values, when the number of iterations reached 3,000, the reward value of SDN-DRLTE algorithm suddenly increased, indicating that the algorithm can more effectively cope with complex and changing network environments. In summary, combining the technical characteristics of DRL and SDN, the computer network traffic control algorithm has shown significant performance improvement and advantages in practical applications. This model is not only suitable for specific network environments, but also can be extended to

various complex network environments and traffic patterns due to its adaptive learning ability based on DRL, providing a more efficient and intelligent network traffic management solution.

6 Conclusion

A DRL-based SDN-DRLTE algorithm was developed to address the traffic control in computer networks. The DRL algorithm process was combined with Markov stochastic process decision-making. When the form of policy rules was uncertain, the DRL algorithm was used to select the optimal performance policy. Meanwhile, the offline policy training method of Off policy random action exploration was adopted. SDN was introduced to build SDN architecture to enhance the flexibility of the network. The DDPG was taken to solve the continuous control problem in computer networks, establishing a DRL-based SND flow control model to achieve flow scheduling and control. Finally, the SDN-DRLTE flow control algorithm was obtained. According to the research results, in terms of algorithm performance, the rate drop of SDN-DRLTE was significantly smaller than other methods. The average dropped to the lowest when the rate requirement reached 50Mbps. Compared with bother algorithms, SDN-DRLTE significantly reduced the average end-to-end latency. As for the practical application effect, the average reward of SDN-DRLTE was optimized by 10%. When the number of iterations reached around 2,800, the convergence speed and reward value increased significantly, and continued to maintain high reward values in subsequent iterations. The research results indicate that SDN-DRLTE algorithm far exceeds traditional algorithms in terms of computational speed, latency, and practical application effectiveness, and has stronger learning and adaptability capabilities. However, there is still a lack of consideration for algorithm training under changes in algorithm inputs. In the future, fixing the algorithm inputs can be eliminated to achieve a more comprehensive flow control effect. Extensive experimental results in different network environments, traffic loads, and application scenarios have shown that the SDN-DRLTE algorithm has stronger learning and adaptability in terms of computational speed, latency, and practical application effectiveness. When the input of the algorithm changes, such as changes in network topology, traffic patterns, or external environmental interference, the algorithm performance may be affected. Subsequently, the robustness of the algorithm can be enhanced, and its strategy can be quickly adjusted to cope with the changes to achieve a more comprehensive traffic control effect.

Funding

The research is supported by: This paper is a fund project of Guangxi Eco-Engineering Vocational & Technical College, "Virtual Simulation Resource Management Platform Construction and operation and maintenance Strategy Research" (No. 2024KY32).

References

[1] Cisco. Cisco Visual Networking Index: Forecast and Trends, 2019-2024, 2020. https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-741490.html

[2] Peter P. Groumpos. A critical historic overview of artificial intelligence: Issues, challenges, opportunities, and threats. Artificial Intelligence and Applications, 1(4):197-213, 2023. https://doi.org/10.47852/bonviewAIA3202689

[3] Reddy SaiSindhuTheja, and Gopal K. Shyam. A machine learning based attack detection and mitigation using a secure SaaS framework. Journal of King Saud University-Computer and Information Sciences, 34(7):4047-4061, 2022. https://doi.org/10.1016/j.jksuci.2020.10.005

[4] Matjaž Gamsd, and Tine Kolenik. Relations between electronics, artificial intelligence and information society through information society rules. Electronics, 10(4):514, 2021. https://doi.org/10.3390/electronics10040514

[5] Chang Peng, and Chengcheng Xu. Combined variable speed limit and lane change guidance for secondary crash prevention using distributed deep reinforcement learning. Journal of Transportation Safety & Security, 14(12):2166-2191, 2022. https://doi.org/10.1080/19439962.2021.2011810

[6] Aberkane, Sabrina, and Mohamed Elarbi-Boudihir. Deep reinforcement learning-based anomaly detection for video surveillance. Informatica, 46(2), 2022. https://doi.org/10.31449/inf.v46i2.3603

[7] Huiling He. Automatic network traffic scheduling algorithm based on deep reinforcement learning. Informatica, 48(22), 2024. https://doi.org/10.31449/inf.v48i22.6943

[8] Masaki Morimoto, Kai Fukami, Kai Zhang, and Koji Fukagata. Generalization techniques of neural networks for fluid flow estimation. Neural Computing and Applications, 34(5):3647-3669, 2022. https://doi.org/10.1007/s00521-021-06633-z

[9] Surjeet Dalal, Bijeta Seth, Vivek Jaglan, Meenakshi Malik, Surbhi, Neeraj Dahiya, Uma Rani, Dac-Nhuong Le, Yu-Chen HuAuthors Info, and Claims. An adaptive traffic routing approach toward load balancing and congestion control in Cloud–MANET ad hoc networks. Soft Computing, 26(11):5377-5388, 2022. https://doi.org/10.1007/s00500-022-07099-4

[10] Jiewu Leng, Weinan Sha, Zisheng Lin, Jianbo Jing, Qiang Liu, and Xin Chen. Blockchained smart contract pyramid-driven multi-agent autonomous process control for resilient individualised manufacturing towards Industry 5.0. International Journal of Production Research, 61(13):4302-4321, 2023. https://doi.org/10.1080/00207543.2022.2089929

[11] Tariq Mahmood, Jianqiang Li, Yan Pei, Faheem Akhtar, Suhail Ashfaq Butt, Allah Ditta, and Sirajuddin Qureshi. An intelligent fault detection

approach based on reinforcement learning system in wireless sensor network. The Journal of Supercomputing, 78(3):3646-3675, 2022. https://doi.org/10.1007/s11227-021-04001-1

[12] Fuat Kosanoglu, Mahir Atmis, and Hasan Hüseyin Turan. A deep reinforcement learning assisted simulated annealing algorithm for a maintenance planning problem. Annals of Operations Research, 339(1):79-110, 2024. https://doi.org/10.1007/s10479-022-04612-8

[13] Hepeng Li, and Haibo He. Learning to operate distribution networks with safe deep reinforcement learning. IEEE Transactions on Smart Grid, 13(3):1860-1872, 2022. https://doi.org/10.1109/TSG.2022.3142961

[14] Shanu Bhardwaj, and S. N. Panda. Performance evaluation using RYU SDN controller in software-defined networking environment. Wireless Personal Communication, 122(1):701-723, 2022. https://doi.org/10.1007/s11277-021-08920-3

[15] Degan Zhang, Lixiang Cao, Haoli Zhu, Ting Zhang, Jinyu Du, and Kaiwen Jiang. Task offloading method of edge computing in internet of vehicles based on deep reinforcement learning. Cluster Computing, 25(2):1175-1187, 2022. https://doi.org/10.1007/s10586-021-03532-9

[16] José de Jesús Rugeles Uribe, Edward Paul Guillen, and Leonardo S. Cardoso. A technical review of wireless security for the internet of things: Software defined radio perspective. Journal of King Saud University-Computer and Information Sciences, 34(7):4122-4134, 2022. https://doi.org/10.1016/j.jksuci.2021.04.003

[17] Song Inseok, Prohim Tam, Seungwoo Kang, Seyha Ros, and Seokhoon Kim. DRL-based backbone SDN control methods in UAV-assisted networks for computational resource efficiency. Electronics, 12(13):2984, 2023. https://doi.org/10.3390/electronics12132984

[18] Linqiang Huang, Miao Ye, Xingsi Xue, Yong Wang, Hongbing Qiu, and Xiaofang Deng. Intelligent routing method based on dueling DQN reinforcement learning and network traffic state prediction in SDN. Wireless Networks, 30(5):4507-4525, 2024. https://doi.org/10.1007/s11276-022-03066-x

[19] Mohamed Escheikh, and Wiem Taktak. Online QoS/QoE-driven SFC orchestration leveraging a DRL approach in SDN/NFV enabled networks. Wireless Personal Communications, 137(3):1511-1538, 2024. https://doi.org/10.1007/s11277-024-11389-5

[20] Hernández-Chulde C, Casellas R, Martínez R, Martínez R, Vilalta R, and Muñoz R. Experimental evaluation of a latency-aware routing and spectrum assignment mechanism based on deep reinforcement learning. Journal of Optical Communications and Networking, 15(11):925-937, 2023. https://doi.org/10.1364/JOCN.499343