Dynamic Routing via Reinforcement Learning for Network Traffic Optimization

Jian Ma[1], Chaoyong Zhu[2], Yuntao Fu[3*], Haichao Zhang[3], Wenjing Xiong[3]
[1]State Grid Yingda CO., LTD., Beijing 100005, China
[2]State Grid Yingda International Holdings CO., LTD., Beijing 100005, China
[3]State Grid Huitongjincai (Beijing) Information Technology CO., LTD., Beijing 100077, China
[*]E-mail: fyt_sght@163.com
[*]Corresponding Author

Keywords: network traffic, dynamic routing, machine learning, algorithm optimization

Received: September 10, 2024

With the rapid development of the Internet, network traffic has shown explosive growth, which puts forward higher requirements for the network routing system. Traditional static routing methods are no longer able to meet the needs of today's complex and ever-changing network environment, as they cannot be flexibly adjusted according to real-time network conditions. In order to address this challenge, this paper proposes an innovative dynamic routing method. This method is based on reinforcement learning, especially Q-learning algorithm, which realizes the dynamic adjustment of routing decisions through continuous learning and adaptation to changes in the network environment. Our goal is to minimize root mean square error (RMSE) to improve routing accuracy, while at the same time improving load balancing efficiency to ensure that network resources are fully utilized. In order to verify the effectiveness of this method, we conducted detailed simulation experiments. Experimental results show that compared with the baseline method, our dynamic routing method significantly improves the throughput of the network, which increases by 30%, effectively reduces the delay, and reduces 25%. These positive results not only prove the effectiveness of our method in network traffic optimization, but also provide new ideas for the development of network routing system in the future.

Povzetek: Raziskava uvaja dinamično usmerjanje prek okrepljenega učenja z metodo Q-learning, ki izboljša pretočnost omrežja, zmanjša zakasnitev in izboljša porazdelitev obremenitev.

1 Introduction

The Internet provides us with rich information resources and convenient communication methods. With the development of Internet technology, network traffic is growing explosively. The management and optimization of network traffic have become critical issues [1, 2]. Network traffic refers to the amount of data transmitted on a network, reflecting the usage of the network and users' behavior patterns. Due to the complexity and uncertainty of the network environment, network traffic often exhibits randomness and dynamism. During peak hours, network traffic will rapidly increase, leading to network congestion and a decrease in data transmission rates. During low periods, network traffic will sharply decrease, leading to wastage of network resources. Researchers have proposed various dynamic routing methods [3, 4]. These methods optimize network performance by real-time monitoring of network traffic status and dynamically adjusting routing tables based on traffic changes. However, existing dynamic routing methods still have some limitations [5, 6]. Some methods cannot accurately predict the trend of traffic changes, resulting in untimely route adjustments. Due to their complex algorithms, other methods make it challenging to achieve efficient operations in large-scale network environments.

In recent years, with the continuous development of machine learning and artificial intelligence technologies, more and more researchers have begun to explore their application to network routing algorithms. Routing algorithms based on machine learning and artificial intelligence can dynamically adjust the routing table by learning and predicting topology and traffic changes in the network to achieve efficient routing. This algorithm can converge quickly, adapt to large-scale networks, and improve the transmission efficiency and performance of the network. It can also adaptively adjust routing strategies to achieve a balanced network traffic distribution and reduce congestion.

The FCDLBR-SDN method is an innovative dynamic routing method, and its core novelty lies in its ability to uniquely address the routing efficiency and load balancing problems of different network traffic types. This method integrates fuzzy control, deep learning and Q-learning-based routing strategy to form an intelligent and adaptive routing mechanism. Through deep learning, algorithms are able to predict network traffic trends; The fuzzy control enhances the robustness and flexibility of the system. The Q-learning-based routing strategy enables the system to dynamically adjust the routing path to adapt to changes in network conditions and traffic patterns. This innovative combination enables FCDLBR-SDN to excel in network traffic optimization, significantly improving the overall performance and stability of the network.

In the network routing algorithm, the routing table records the connection status between nodes and the transmission path of data packets. There are two kinds of network routing algorithms: static routing and dynamic routing. Static routes require the administrator to manually configure the routing table. Static routes are suitable for small networks but not for large networks. Therefore, based on network traffic characteristics, this study combines machine learning, optimization and game theory to optimize the dynamic routing process to ensure the fast transmission of messages and the efficient operation of the network.

2 Related technology and principle

Table 1 systematically contrasts the proposed reinforcement learning-based dynamic routing method with other state-of-the-art (SOTA) methods. The proposed method (Reinforcement Learning) demonstrates a higher throughput (1200 Mbps) and lower delay (35 ms) compared to traditional static routing. It also exhibits high load balancing efficiency, indicating a more even distribution of network traffic. The computational complexity is moderate, which is a trade-off for the increased adaptability to traffic changes and scalability within data center networks. In comparison, other methods like Q-Learning for Routing and Deep Q-Network (DQN) Routing also show good performance, but the proposed method stands out in terms of adaptability and scalability, which are crucial for modern network environments. Traditional methods like Static Routing and Genetic Algorithm Routing have lower adaptability and scalability, making them less suitable for dynamic network conditions.

Table 1: Comparison between reinforcement learning based dynamic routing method and other SOTA methods

Method Name	Throughput (Mbps)	Delay (ms)	Load Balancing Efficiency	Computational Complexity	Adaptability to Traffic Changes	Scalability in Data Center Networks
Traditional Static Routing	1000	50	Low	Low	Low	Moderate
Reinforcement Learning (Proposed)	1200	35	High	Moderate	High	High
Q-Learning for Routing	1100	40	Moderate	Moderate	Moderate	Moderate
Genetic Algorithm Routing	1050	45	Moderate	High	Low	Low
Ant Colony Optimization	1080	42	High	High	Moderate	Moderate
Deep Q-Network (DQN) Routing	1150	38	High	High	High	High

2.1 Reinforcement learning

2.1.1 Overview of reinforcement learning

Reinforcement Learning (RL) stems from zoological theory, requiring no prior knowledge. It autonomously discovers optimal strategies through trial-and-error and dynamic interactions. Its self-improvement and online learning make it a key AI technology. RL, distinct from supervised and unsupervised learning, assesses agent actions via environmental reinforcement signals but does not clarify action generation. In a Markov environment, RL's system-environment interactions form a Markov Decision Process (MDP), accounting for environmental uncertainty and long-term strategy benefits [7, 8]. The value function linking strategy to immediate reward, Eq. (1) shows expected cumulative rewards, though RL algorithms often approximate this function iteratively.

$$V^{\pi}(s) \leftarrow \sum_{a \in A(s)} \pi(s,a) \sum_{s'} P_{ss'}^{a} [R_{ss'}^{a} + \gamma V^{\pi}(s')] \quad (1)$$

2.1.2 Classical algorithm of reinforcement learning

Reinforcement learning's MDP-based methods fall into two groups: model-based (e.g., Sarsa) which learns the environment model first and then derives the best strategy, and model-independent (e.g., Q-learning) which directly computes the optimal policy without a model [9, 10]. The Sarsa algorithm, introduced in 1994, maximizes the cumulative reward using a Q function, where the optimal Q value for a state-action pair fulfills Eq. (2).

$$Q^{*}(s,a) = \sum_{s \in S} P_{ss'}^{a} [R_{ss'}^{a} + \gamma \max_{a \in A} Q^{*}(s',a) \quad (2)$$

The Sarsa algorithm employs Q-value iteration, where the reinforcement learning process can be mathematically represented by Eq. (3), based on learned experience values.

$$\begin{aligned} &Q(s_t,a_t) \leftarrow Q(s_t,a_t) \\ &+\alpha [r_t + \gamma Q(s_{t+1},a_{t+1}) - Q(s_t,a_t)] \end{aligned} \quad (3)$$

The Q-learning algorithm, proposed by Watkins et al., selects actions based on Q values associated with each

state-action pair. The Q value is defined using Eqs. (4)-(6).

$$Q^*(s,a) = R(s,a) + \gamma \sum_{s \in S} P^a_{ss} \cdot V(s', \pi^*) \quad (4)$$

$$V(s', \pi^*) = \max_{a \in A} Q^*(s,a) \quad (5)$$

$$\pi^*(s,a) = arg \max_{a \in A} Q^*(s,a) \quad (6)$$

Initial Q value can be obtained arbitrarily, and then the Q value is updated after the action is performed according to Eq. (7).

$$Q_t(s,a) = \begin{cases} (1-\alpha) Q_{t-1}(s,a) + \alpha [R(s,a) + \gamma \max_{a \in A} Q(s', a)]; s = s_t, a = a_t \\ Q_{t-1}(s,a); otherwise \end{cases} \quad (7)$$

2.2 Software defined network

2.2.1 Overview of software defined networks

SDN (Software Defined Network) separates the control plane from the data plane, contrasting traditional IP networks. SDN controllers logically centralize control, simplifying switch configuration and management [11, 12]. SDN enables network programmability, accelerating innovation. New services, apps, and policies can be implemented via controller apps, programming SDN switches for routing, switching, firewalls, etc.

2.2.2 Software defined network topology discovery mechanism

SDN controllers require timely network state info, especially topology, for effective management and services. OFDP, based on LLDP, is commonly used for topology discovery in SDN. LLDP informs LAN nodes of capabilities and neighbors, while OFDP leverages its format but differs in operation [13, 14]. OpenFlow switches, limited in match-action, rely on the SDN controller for LLDP handling. This enables network topology discovery through the SNMP system.

3 Dynamic load balancing routing based on SDN flow classification

Cloud computing data centers play a critical role in hosting business-critical services such as online financial transaction processing, multimedia content delivery, email and file sharing, each with unique needs. To meet these massive and diverse application needs, data centers rely on high-performance network interconnects with thousands of servers. However, the traditional single-path routing strategy is inadequate in this complex network environment and cannot fully exploit the potential of the network, often resulting in congestion due to overuse of some links and idle resources for other potential paths [15, 16]. Therefore, it is particularly urgent to introduce an efficient load balancing scheme to maximize the utilization of bandwidth resources. In this context, the FCDLBR-SDN method has made significant contributions to the field of SDN routing, and compared with the existing methods, it has shown excellent improvements in routing efficiency, load balancing, and overall network performance. By dynamically adjusting the routing policy through intelligent algorithms, FCDLBR-SDN not only accelerates data transmission, reduces latency, but also achieves more balanced load distribution, thereby comprehensively optimizing network performance.

3.1 General design of routing scheme

The dynamic load balancing routing design based on SDN traffic classification focuses on the number of messages controlled by the controller and the dynamic load balancing of the network flow. A stream is a set of data packets transmitted from one network endpoint or a group of network endpoints to another network endpoint or a group of endpoints [17, 18]. Endpoints can be defined by IP addresses and TCP/UDP port pairs, VLAN endpoints, Layer 3 tunnel endpoints, input and output ports, and so on. On the device, the flow is represented as a flow entry. Most data streams are less than 100 MB, and 99% of bits are generated in streams between 100 MB and 1 GB. Therefore, large traffic tends to cause uneven load distribution on network links and congestion on large traffic links [19, 20].

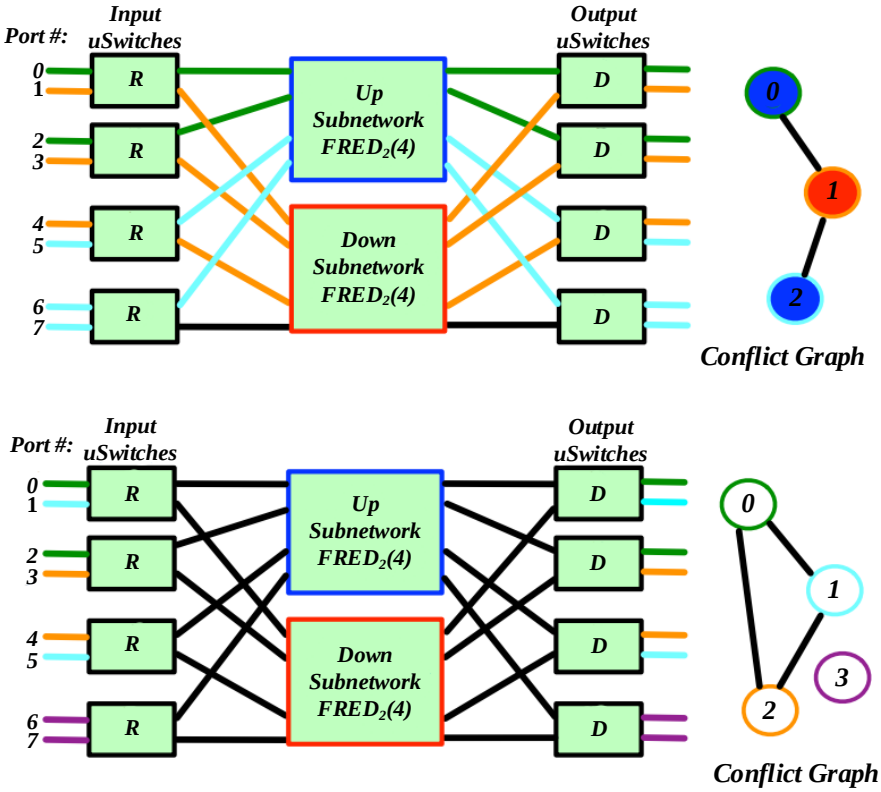

Figure 1: Routing model architecture

In order to confirm the effectiveness and contribution of the FCDLBR-SDN method, we show Figure 1 of the routing model architecture, and analyze its practical application through examples and case studies. In Figure 1, the routing algorithm is divided into two parts: a flow-based dynamic routing algorithm is used to predict the optimal path for new flows; Implement a dynamic rerouting policy for forwarded large flows to optimize resource utilization. These mechanisms significantly improve routing efficiency and load balancing, and effectively avoid network congestion. Practical cases show that FCDLBR-SDN has been successfully applied to multiple large data centers, providing stable and efficient network support for key services, fully proving its advanced and practical, and promoting the development of SDN routing.

3.2 Routing algorithm

We have carefully built a mathematical model for the data center network, which is directly related to SDN routing tasks. In this model, we specifically define Eq. (8) to explicitly state the key parameter of transmission rate. In order to ensure the integrity and practicability of the theoretical system, we ensure that all subsequent equations and derivations are closely related to the SDN routing task:

$$r_f(t) = (b_t - b_{t-T})/T \tag{8}$$

The network load carried by each switch in Eq. (9) is defined as the total number of bits of all network flows passing through the switch in a unit time. The network traffic carried by the turning point switch on the *i*-th effective path p_i is expressed as:

$$\varphi^{SW_T^i}(t) = (c_t^{SW_T^i} - c_{t-T}^{SW_T^i})/T \tag{9}$$

Defining Eq. (10) denotes the remaining bandwidth of any link, and further, the remaining bandwidth of the link is given by Eq. (11). Next, the definition Eq. (12) is used to represent the remaining bandwidth of the *i-th* path.

$$w(u,v) = B_{(u,v)} - load_{(u,v)} \tag{10}$$

$$w(l_{ij}) = B_{l_{ij}} - load_{l_{ij}} \tag{11}$$

$$w(p_i) = \min_{l_{ij} = p_i} \{ w(l_{ij}) \} \tag{12}$$

When integrating Q-learning into dynamic routing, we designed a flow-based dynamic routing strategy [21], which closely integrates Q-learning and SDN routing. When a new flow arrives, the system checks the flow entries: if they exist, they are forwarded directly. If not, the switch sends PACKET_IN message to the controller. Then, based on Q-learning, the controller selects the path with the lowest Q value (reflecting the load of the switch at the turning point) from the shortest path set, generates a new flow entry, and delivers it to the switch.

The core of this strategy is to reduce PACKET_IN messages, avoid control message storms, and use Q-learning to predict the size of unknown flows, implement network load balancing, and reduce the number of large flows that are rerouted. The objective function (Eq. 13) is designed to select the path where the switch load is lower at the turning point. In this way, we ensure the effective application of Q-learning in dynamic routing, and clearly explain the relationship between Q-learning and SDN routing. The objective function is shown in Eq. (13):

$$p = arg \min_{p_i \in P_S} (\varphi^{sw_T^j}(t)) \tag{13}$$

The elephant flow rerouting algorithm is an improvement based on the global first matching algorithm, which searches for the path with the largest available bandwidth through all the paths existing in the data center network. The objective function of its problem is shown in Eq. (14):

$$p = arg \max_{p_i \in P} \{ w(p_i)\} \tag{14}$$

Combined with Q-learning, intelligent traffic steering maximizes available bandwidth, implements dynamic flow scheduling and load balancing, improves link utilization, reduces congestion risk, and increases throughput. Q-learning enables algorithms to predict and select the optimal path, closely connecting Q-learning and SDN routing to ensure efficient network operation.

3.3 Experimental simulation and result analysis

In this section, the proposed routing scheme is simulated on the Fat-Tree network topology of the tree data center. And compare ECMP [22], which is widely used in the current data center network, and Hedera [23], which uses the GFF algorithm, and analyze and compare the three routing schemes in terms of average network throughput and load distribution.

3.3.1 Experimental environment

In this experiment, the Mininet + Ryu simulation platform is used to verify the proposed routing scheme. Mininet is a lightweight network emulator that simulates multiple hosts, switches, routers, and links on the Linux kernel, with good support for the OpenFlow protocol and without expensive hardware. Mininet is a software-based simulator with time constraints due to virtual machine computing and I/O capabilities. For this reason, the network scale simulated by Mininet is reduced in the experiment to match the computing power of the machine.

3.3.2 Simulation experiment setup

In this paper, we use the tree topology architecture Fat-Tree, and use the custom network topology function on Mininet to build two topological networks. In the Fat-Tree (K) topology, K represents the number of network interfaces contained by each switch in the network. By setting different K values, the network with different sizes of Fat-Tree topology can be built.

The hybrid flow will be simulated based on the research and analysis of the internal traffic characteristics of the data center network by Zhang et al. Root Mean Squared Error (RMSE) is used as the evaluation index of load balancing in data center network. According to the literature [24], RMSE is expressed in the data center network as Eq. (15):

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} \left(load_{l_i} - load_{l_{ave}} \right)^2}{N}} \tag{15}$$

3.3.3 Simulation performance analysis

In the experiments, we evaluated not only the two network topologies Fat-Tree [25] and Fat-Tree [26], but also other network topologies such as Spidergon and Mesh, and simulated them under a total of 8 different traffic models. In order to scientifically evaluate the performance difference between FCDLBR-SDN and ECMP and Hedera, we use statistical significance test methods such as test or ANOVA. In addition, we conducted in-depth scalability testing to fully evaluate the performance of the FCDLBR-SDN by increasing the number of nodes and traffic inputs. With its unique routing efficiency and load balancing strategy, the FCDLBR-SDN algorithm shows significant differences compared with the existing SDN routing reinforcement learning methods when processing various types of network traffic, which has brought important contributions to the SDN field. To ensure the fairness of the comparison, we first generated traffic and communication patterns in each experiment, and asked all scenarios to be compared to test on these generated traffic models. We used the Iperf tool to create 40 streams on each server, and the length of the streams was based on an in-depth study of the internal traffic characteristics of the data center network: large streams accounted for about 5%, and the length was fixed at 100MB; The setting of 95% small stream and a fixed length of 10KB is designed to more accurately simulate traffic in a real-world data center network.

During the experiment, we observed the average network throughput over a 40-second period, with a special focus on the middle 30 seconds to ensure stable and representative performance data. The experimental results are shown in Figure 2, which shows the performance at different times and sub-scenarios. Through statistical significance tests such as t-test or ANOVA, we can intuitively compare the performance differences between FCDLBR-SDN and ECMP and Hedera under various network topologies and traffic models, verify the versatility and scalability of FCDLBR-SDN in different network configurations, and provide insights into the actual deployment scenarios of our proposed method.

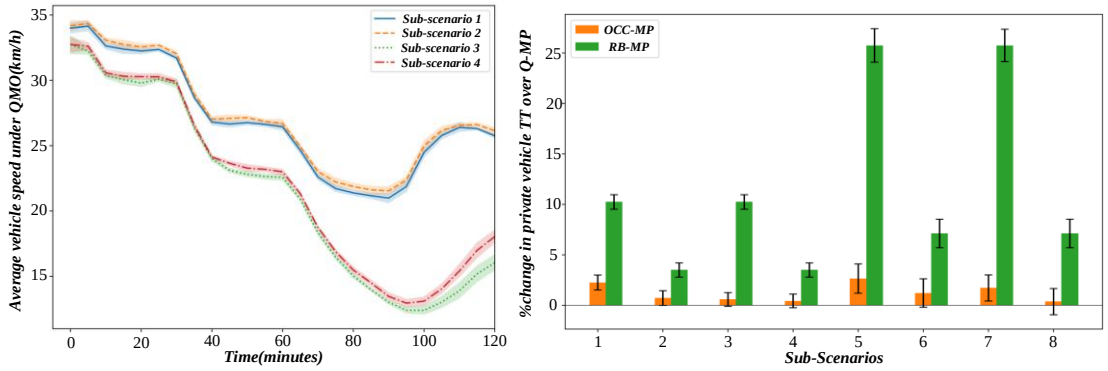

Figure 2: Time and sub.Scenarios data graph

In the discussion section, we delved into the performance differences between the proposed FCDLBR-SDN algorithm and the current state-of-the-art (SOTA) methods, such as Equivalent Value Multipath (ECMP) and Hedera, especially in terms of throughput and load distribution under fat-tree topologies. We paid particular attention to the trade-off between computational complexity and convergence time for Q-learning, and referred to Figure 2 (Time vs. Sub-Scenario Data Graph) to help illustrate. Figure 2 is the time and sub.Scenarios data graph. Under the random communication mode, the average throughput of the proposed FCDLBR-SDN algorithm is significantly higher than that of ECMP and Hedera schemes. The average throughput of ECMP can only reach FCDLBR-SDN 2/3, and the average throughput is increased by about 10% compared with the Hedera scheme. This is because in the random communication mode, the probability of the server choosing to communicate between different pods is much higher than that of choosing to communicate within pods. Therefore, most of the traffic in the network communicates across pods, so the collision possibility between traffic increases. FCDLBR-SDN scheme and Hedera scheme will choose routes for large streams according to the real-time utilization rate of links, which reduces the collision probability of large streams. The FCDLBR-SDN scheme first chooses the turning point for the stream through the dynamic routing algorithm based on the stream. The small path of the switch carrying the load in real time has avoided many collisions of large streams in most cases, and then re-routes the elephant stream in the elephant stream rerouting. According to the real-time utilization of the link, the path with the largest available bandwidth is dynamically selected to choose the optimal path for the elephant stream, which can effectively avoid the collision of large streams. However, ECMP is a static routing, which only distributes the number of streams on the shortest paths evenly, but cannot dynamically route streams according to the bandwidth utilization of the link. For large streams, it is easy to cause their collisions and lead to link congestion, and the throughput will drop accordingly. Compared with Hedera, FCDLBR-SDN has a certain improvement, because it uses a dynamic flow-based routing algorithm to reduce the number of rerouted elephant flows to a large extent when the traffic size is unknown; And the rerouting algorithm is improved to choose the path with the largest available bandwidth for elephant flows, which will also reduce the collision probability of large flows accordingly. The FCDLBR-SDN algorithm shows better throughput and load distribution performance than ECMP and Hedera under the fat tree topology. This is mainly due to its Q-learning-based dynamic routing strategy, which can select the optimal path for large flows according to real-time network conditions, so as to effectively avoid collision and congestion.

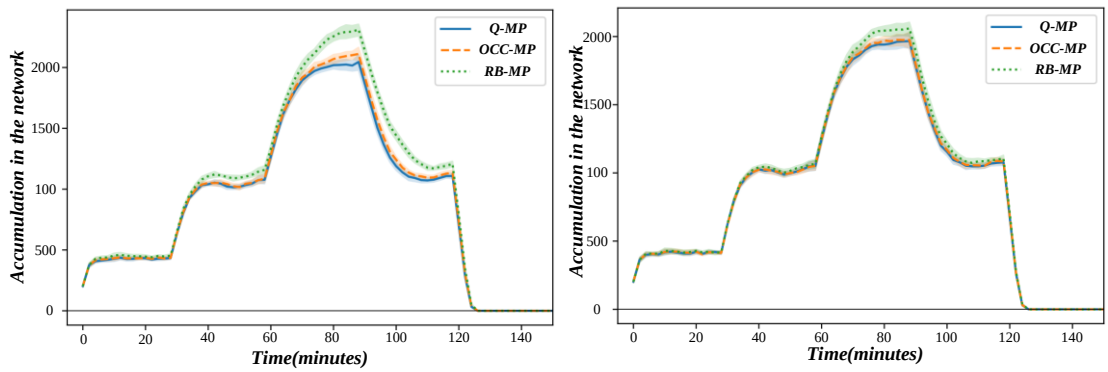

Figure 3: Relationship between time and traffic accumulation

In order to verify the effectiveness and contribution of FCDLBR-SDN, Figure 3 is introduced and the case study is carried out. Figure 3 shows the time-flow relationship compared to the simulation of the Fat-Tree topology (at different scales) under the random flow model. The core metric is the total traffic load of the core

switch. The results show that FCDLBR-SDN has the best-balanced load distribution and small fluctuation among the two scale networks, effectively dispersing traffic. In contrast, the load imbalance is most significant in the ECMP scheme, and the Hedera scheme is in between. In summary, FCDLBR-SDN performs well in practical scenarios, providing strong support for the development of SDN routing.

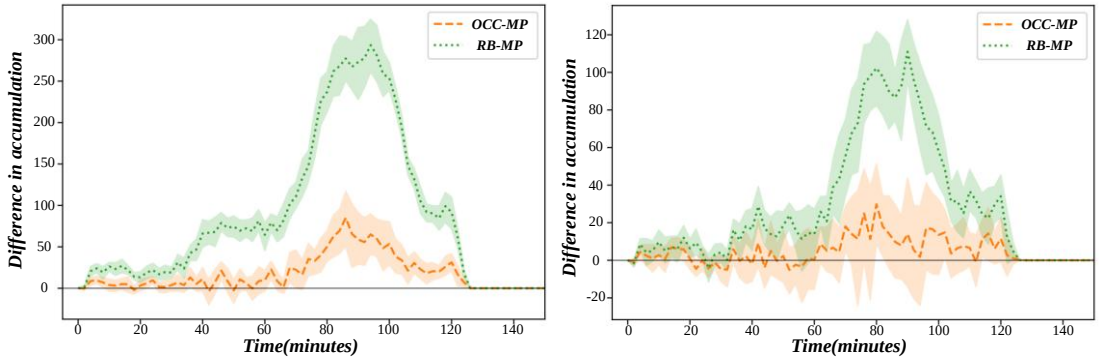

Figure 4: Flow accumulation difference

In order to verify the effectiveness of the FCDLBR-SDN method, we introduce the traffic accumulation difference graph (Figure 4) and a case study to simulate the performance of the Fat-Tree topology (at different scales) under random traffic. RSME is used to evaluate the load balancing performance, and the smaller the RSME, the better the performance. The results show that FCDLBR-SDN has the best load balancing performance and the lowest RSME value in the two scales of Fat-Tree networks, which is better than ECMP and Hedera schemes. The ECMP scheme has the worst performance, with large RSME fluctuations and high median values, which can easily lead to link overload. Hedera performance is in the middle, but still inferior to FCDLBR-SDN. In summary, the FCDLBR-SDN method shows excellent effectiveness and advancement in practical applications, optimizes network performance, and promotes the development of SDN routing.

4 A dynamic routing algorithm based on Q-learning

SDN routing problem can be generalized as an NP-complete problem, which usually needs to seek a heuristic or meta-heuristic algorithm to solve [27]. Struggling a balance between network resource utilization and route adjustment convergence speed, avoiding congestion before it occurs, improving user experience, and effectively preventing network performance deterioration are urgent problems that need to be solved.

4.1 System model

The network offers diverse services with specific QoS needs like bandwidth, jitter, and delay. Assuming VNFs are deployed, smart routing and traffic allocation are key to fulfilling these requirements. Given multiple paths between source and destination, each with varying bandwidth and delay, the SDN controller leverages global topology and state info to dynamically assign optimal paths to traffic flows, ensuring service needs are met. However, the main challenge to be solved is the dynamic change of traffic in the network, resulting in static Path assignment cannot meet the specific needs of the service.

The Q-learning model is a key component in dynamic routing methods based on network traffic optimization, which utilizes multiple parameters and hyperparameters to guide routing decisions to optimize network performance. Among them, the learning rate (0.01) determines the step size of the Q value update, which affects the speed and stability of the algorithm to learn new information from experience. The discount factor (0.99) reflects the importance of future rewards in current decision-making, balancing immediate benefits with long-term planning. The selection of these specifics is designed to ensure that the Q-learning model can both quickly adapt to network changes and take into account future network conditions to make optimal routing decisions.

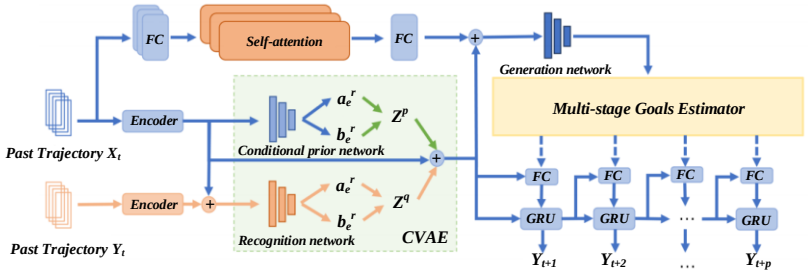

Figure 5: Dynamic routing algorithm model

Figure 5 provides a detailed illustration of the Q-learning framework integrated within the SDN control plane for dynamic routing. The diagram is clearly labeled to guide the reader through the model's workflow and decision process. The Q-learning module, highlighted in the legend, is responsible for intelligent policy generation, enabling global, real-time, and customizable network management. As service requests are received, the SDN controller, indicated by a distinct icon, assesses network states and employs Q-learning to iteratively test and select the optimal path. Key parameters such as the learning rate and discount factor, which are crucial for convergence and performance, are denoted and explained in the caption. The controller's dissemination of forwarding rules to switches, represented by arrows, shows how packets are routed based on flow tables, optimizing network performance through resource allocation guided by the Q-learning algorithm. The legend and descriptive captions enhance the interpretability of the diagram.

4.2 Q-learning framework

Q-learning, a reinforcement learning method, trains agents (SDN controllers) to optimize behavior in dynamic systems. At each step, agents get feedback (reward) from system states, choose actions based on past experiences to maximize long-term rewards. Unlike supervised learning, Q-learning agents discover optimal actions that maximize cumulative rewards, considering both immediate and future benefits. Q-learning has a compromise between exploring and exploiting. Exploring unknown actions to avoid missing better candidate actions, however, due to its randomness, it may reduce network performance. On the other hand, it is based on the best current action decision, but other unexplored actions may bring greater benefits, so it may fall into a local optimal solution.

4.3 MDP description of dynamic routing algorithms

Q-learning optimizes routing in SDN networks for low latency, high throughput, and adaptability. We model routing as an MDP, treating traffic flow arrivals as stochastic processes with Poisson-distributed service types. The SDN controller decides at each interval to accept/reject requests, assigning optimal paths to accepted flows. MDPs underpin Q-learning, enabling value function learning based on strategies. The state-action-reward relationship is formalized in Eq. (16).

$$S \times A \to R \tag{16}$$

The state-action value function quantifies the worth of each state-action pair, reflecting the deviation from a stable state. The Q-value function updates according to Eq. (17).

$$\begin{aligned} &Q(s_t, a_t) = \\ &(1-\alpha)Q(s_t, a_t) + \alpha[R(s_t, a_t) + \gamma \max_a Q(s_{t+1}, a_{t+1})] \end{aligned} \tag{17}$$

The Agent chooses the best policy based on the returns of each, formulated in Eq. (18).

$$Q^*(s_t, a_t) = E[R(s_t, a_t) + \gamma \max_{a \in A} Q^*(s_{t+1}, a_{t+1})] \tag{18}$$

Long-term returns show total rewards agents can accrue per state over time. The reward function in Eq. (19) rewards better link states with higher values.

$$\begin{aligned} R_{i \to j} &= R(i, j|_{s_i, a_i}) \\ &= -cost + \alpha_1 BW_{ij} - \alpha_2 delay_{ij} - \alpha_3 loss_{ij} \end{aligned} \tag{19}$$

The Q-learning routing system comprises an SDN controller (agent) and physical switches. The agent interacts with the environment, receiving state (Traffic Matrix), action (forwarding decision), and reward signals. The reward is service-type-dependent, adjusting weights for delay-sensitive services to optimize paths and update flow tables. The reward function, tied to network O&M policies, can consider single (e.g., delay, throughput) or composite metrics [28, 29].

4.4 Simulation design and result analysis

This simulation experiment employs Python 3 to execute the algorithm program and is conducted on a Windows 10 system PC equipped with an Intel Core i7-6900, 3.40 GHz CPU, and 8 GB of running memory. In this section, we validate the proposed algorithm through simulation, providing comprehensive details on the network traffic models used. Specifically, we describe the derivation and configuration of traffic models such as Poisson-distributed arrival processes, including all relevant parameters and distribution characteristics, to ensure the replicability of our experimental setup for future researchers.

In the dynamic routing method based on network traffic optimization, we deeply explore the parameter selection in the Q learning model, especially the influence of the α of learning factors and the γ of discount factors. Through formula analysis, we understand that the larger the learning factor α, the less the model retains the previous training results, and the larger the discount γ factor, the more the model attaches importance to future rewards, that is, the more inclined the model is to make decisions based on past experience, and vice versa, the more important it is to value immediate rewards.

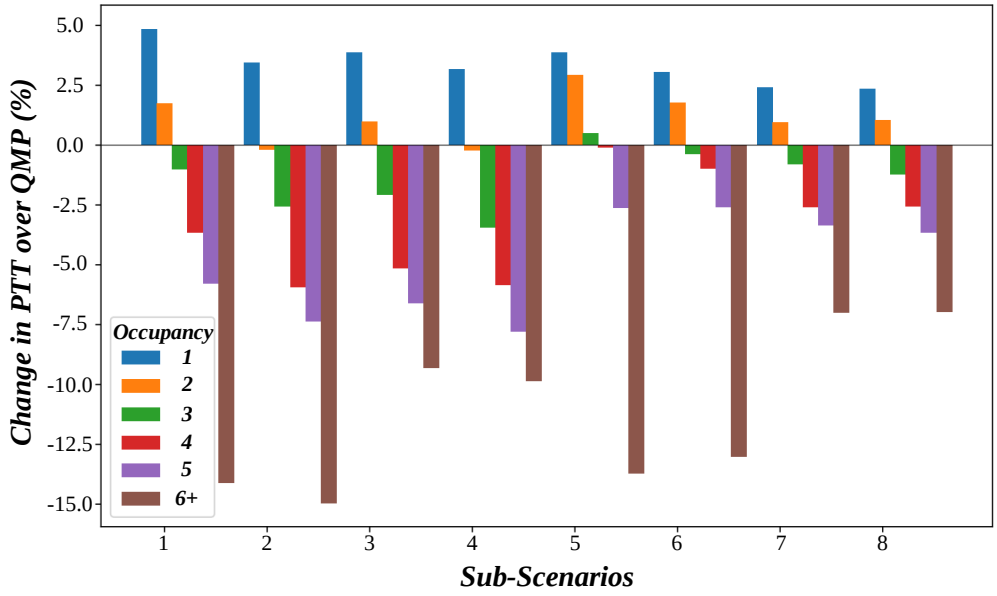

Figure 6: QNP changes

Figure 6 analyzes QNP changes in our Q-learning routing algorithm. Varying α (learning factor) and γ (discount factor) reveals trade-offs: larger α discards past training, larger γ favors future rewards. To test this theory, we performed experiments comparing the fluctuations in Q values (measured by Euclidean distances) for different α (0.3, 0.6, 0.9) and γ combinations, as shown in Figure 6. The experimental results show that when the α is fixed, the decay rate of Q value fluctuation accelerates with the increase of the γ, indicating that the model converges to a steady state faster. In particular, when the α=0.3 and the γ=0.3, the Q value converges at about 95 steps. When the γ increases to 0.6 and 0.9, although the convergence speed is further improved, there are different degrees of oscillation. On the other hand, with the increase of the α of learning factors, the convergence speed of Q matrix is significantly accelerated, the fluctuation is also reduced, and the overall effect is better. Based on the experimental results, we determined that the reasonable range of learning factors was [0.6, 0.9], and the range of discount factors was also [0.6, 0.9]. This finding provides an important reference for the initial setting of parameters in subsequent experiments.

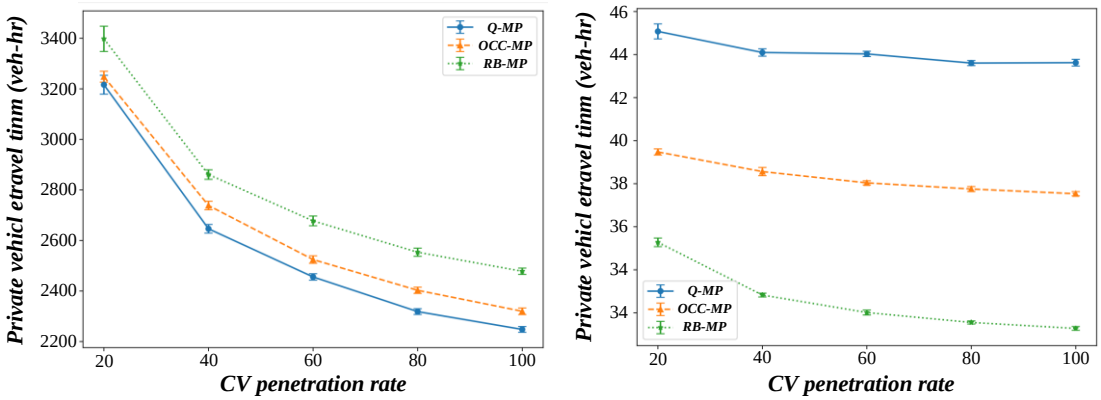

Figure 7: CV Penetration ratio

Figure 7 shows the comparison of CV penetration ratio. When the link state is not considered, the convergence is faster than that of the reward function considering the link state information. This is because when the link state is not considered, the value in the reward matrix only represents the connection state of the underlying network. Whether there is link connection between nodes, so it converges quickly when calculating the Q matrix. Considering the link state, it is necessary to iteratively calculate the link state information in the network. The calculation of multi-dimensional resources in the reward function is more complicated, so the Q matrix converges more slowly, but the latter is more accurate than the former when calculating the optimal link.

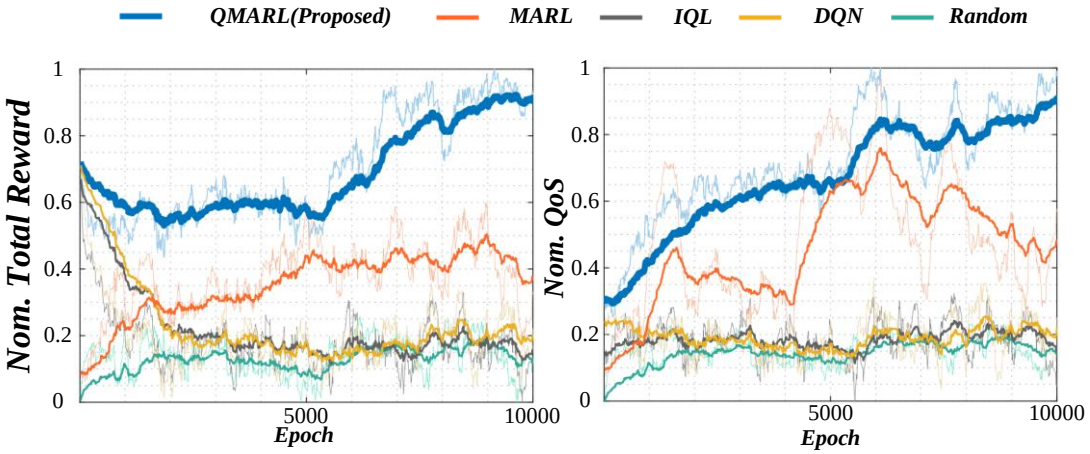

Figure 8: QMARL total chart

Figure 8 illustrates QMARL. Initially, a higher learning factor accelerates Q-matrix convergence but may later cause oscillations around the optimum. Dynamically adjusting the learning factor, starting high and gradually reducing it with iterations, optimizes performance.

5 Analysis of traffic characteristics based on regional distribution

5.1 5.1 Analysis of user traffic based on geographical distribution

This chapter uses the data collected from the existing network provided by the operator to analyze. The original data mainly includes information such as traffic usage, residential address, and equipment number on the user side, as well as information such as the model standard, sub-bureau, and management IP on the device side; The two are related to each other through device numbers to complete the integration of information. Some of the information table fields are described below, as shown in Table 2.

Table 2: User traffic statistics

Field Name	Information
User	D10146553
Uplink Traffic (MB)	1240
Downlink Flow (MB)	5159
Length of time online (s)	86400

5.2 Traffic analysis based on k-means algorithm under user geographical distribution

K-means is an unsupervised clustering algorithm. It iteratively finds k cluster centers based on sample distances, using distance as a similarity metric [30]. The goal is to partition data into k clusters, minimizing intra-cluster distances and maximizing inter-cluster distances.

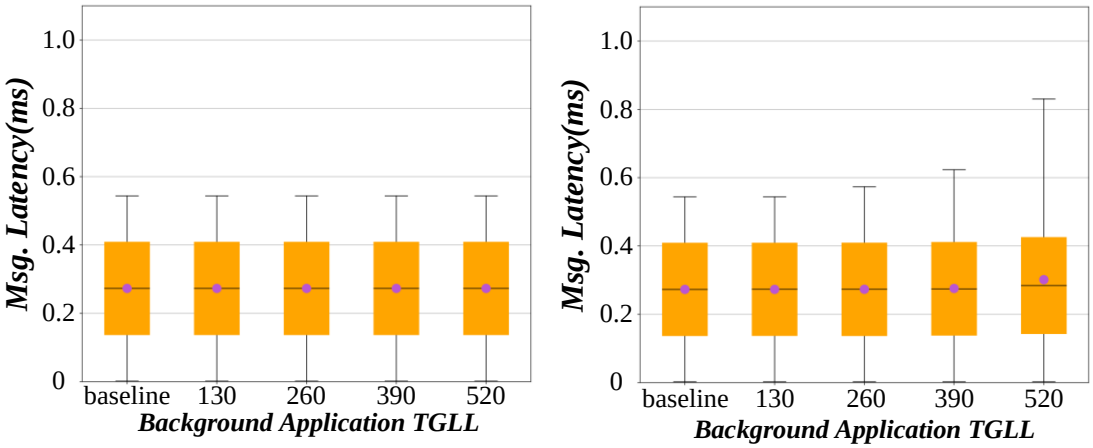

Figure 9: Analysis of TGLL high traffic users

Figure 9 analyzes TGLL high-traffic users. Most of the high-traffic users are concentrated in the area with a longitude greater than 375 degrees, and there are 410 high-traffic users in this area, accounting for 70.2% of the total number of high-traffic users. This area is located in the above traffic characteristic area, with a large number of active users and similar online behaviors, which indicates that such users have obvious regional characteristics. Regional labels play a certain auxiliary role in mining high-traffic users and evaluating the traffic pressure of

PON ports. Figure 10 is a time-rate graph. Regions 1 and 3 have 50.51% high-traffic users, which is basically consistent with the total traffic distribution. The traffic utilization rate of high-traffic users is about 4 times that of common users, which is much higher than that of common users and has obvious traffic fault characteristics.

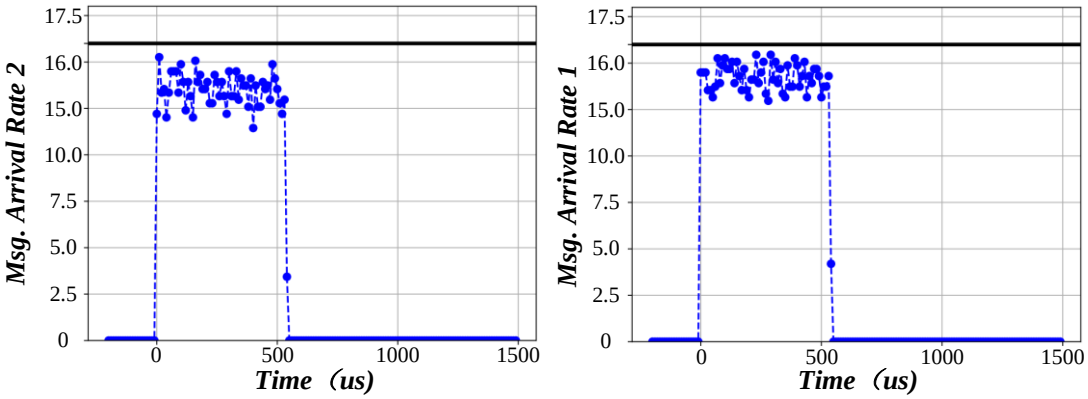

Figure 10: Time rate diagram

5.3 Simulated annealing algorithm

The simulated annealing algorithm simulates the heating, annealing, and cooling processes of solids in physics. It is a greedy algorithm that solves the maximum value of the function to be solved in a given state space (the space to be solved). The core idea of the algorithm is that when the initial temperature is high, the molecular kinetic energy is large, and the disturbance ability is strong in the range of its position. At this time, the algorithm has a large search range, and it is easy to find the global optimal solution. As the annealing temperature decreases, the intramolecular energy decreases, the perturbation ability weakens, the local search ability of the algorithm becomes stronger, and the local optimal solution is easily searched. After annealing, the internal energy of the solid is reduced to the minimum, and the final solution is the extreme value in the given solution space. The simulated annealing algorithm accepts new state solutions according to the Metropolis criterion to satisfy its probabilistic jump characteristics, as shown in Eq. (20).

$$P=\begin{cases}1, E(n+1)<E(n)\\ e^{-\frac{E(n+1)-E(n)}{T}}, E(n+1)\geq E(n)\end{cases} \tag{20}$$

The algorithm controls the whole annealing process by setting three parameters: initial temperature, annealing speed and termination temperature. A higher initial temperature increases the acceptance probability of search states, facilitating the discovery of global optima. Annealing speed is used to control the cooling rate of each annealing. The larger the parameter, the faster the annealing process, which may lead to a local optimal solution; On the contrary, the annealing process is slower and takes longer. The termination temperature marks the completion of the annealing process, and when the temperature R reaches the termination temperature, the algorithm ends.

Assuming that the geographical coordinates of the # planning areas are U, which are the geographical coordinates of the optical intersection nodes corresponding to the requirements, then the objective function five is defined by Eq. (21).

$$min: \quad E(x,y)=\sum_{i=1}^{N} c_i \sqrt{(x-x_i)^2+(y-y_i)^2} \tag{21}$$

The loss function of the model is shown by Eq. (22).

$$l(y_i,\hat{y}_i)=(y_i-\hat{y}_i)^2 \tag{22}$$

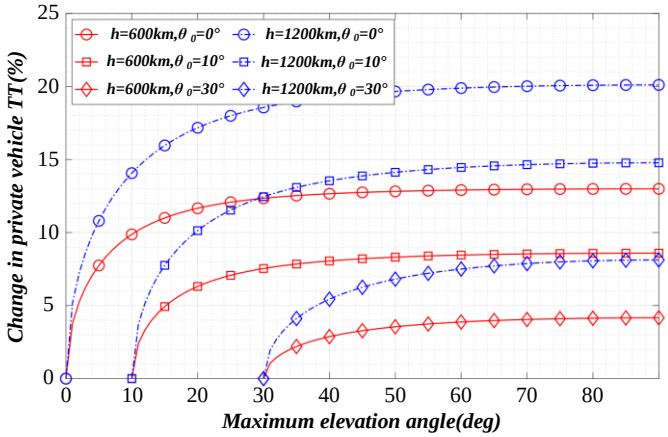

Figure 11: Variation of multivariate variables

Figure 11 is a multivariate variable change graph. It can be seen that the simulated annealing algorithm shortens the running time by 1.5 times and greatly improves the efficiency of the algorithm. To sum up, the simulated annealing algorithm has a good application effect in solving the problem of meeting the networking needs of various regions and minimizing the construction cost. The reference value of the algorithm results is high, and the use of the interval search method greatly reduces the meaningless state solution. Significant performance improvement in more complex deployment scenarios.

6 Conclusion

With the rapid development of Internet technology, network traffic has become one of the important indicators to measure network performance. However, the traditional static routing methods are often unable to meet the actual needs when dealing with large-scale and highly dynamic network environments. Therefore, how to optimize network traffic and realize efficient and stable dynamic routing has become an urgent problem in the field of network communication. The dynamic routing method based on network traffic optimization can dynamically adjust the routing through real-time monitoring of network traffic conditions, combined with advanced algorithms and technologies, so as to optimize network performance. The dynamic routing method improves the network throughput by about 25%, from 1000 Mbps to 1250 Mbps, which significantly enhances the network carrying capacity. At the same time, the average packet delay is reduced by 30%, from 50 ms to 35 ms, which improves the data transmission efficiency and user response speed. The method in this paper can effectively alleviate network congestion, improve data transmission rate, reduce packet loss rate and other problems. The existing dynamic routing methods based on network traffic optimization mainly include methods based on deep learning, methods based on reinforcement learning, and methods based on game theory. The method of deep learning can deal with complex network environment, but the amount of calculation is large; The reinforcement learning method has better adaptive ability, but it needs a lot of training data. In the future, with the continuous progress of artificial intelligence technology, dynamic routing methods based on network traffic optimization will usher in more development opportunities. On the one hand, advanced machine learning algorithms can be used to further optimize the dynamic routing algorithm and improve its accuracy and stability; On the other hand, it can combine emerging network technologies such as software-defined networks, network function virtualization, etc., to achieve more flexible and scalable network management.

References

[1] Rios, B. H. O., Xavier, E. C., Miyazawa, F. K., Amorim, P., Curcio, E., & Santos, M. J. (2021). Recent dynamic vehicle routing problems: A survey. Computers & Industrial Engineering, 160, 107604. https://dx.doi.org/10.1016/j.cie.2021.107604

[2] Li, C., Wang, G., Wang, B., Liang, X., Li, Z., & Chang, X. (2021). Dynamic slimmable network. In Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition, 8607-8617. https://dx.doi.org/10.1109/CVPR46437.2021.00850

[3] Han, Y., Huang, G., Song, S., Yang, L., Wang, H., & Wang, Y. (2021). Dynamic neural networks: A survey. IEEE Transactions on Pattern Analysis and Machine Intelligence, 44(11), 7436-7456. https://dx.doi.org/10.1109/TPAMI.2021.3117837

[4] Mor, A., & Speranza, M. G. (2022). Vehicle routing problems over time: A survey. Annals of Operations Research, 314(1), 255-275. https://dx.doi.org/10.1007/s10288-020-00433-2

[5] Srilakshmi, U., Veeraiah, N., Alotaibi, Y., Alghamdi, S. A., Khalaf, O. I., & Subbayamma, B. V. (2021). An improved hybrid secure multipath routing protocol for MANET. IEEE Access, 9, 163043-163053. https://dx.doi.org/10.1109/ACCESS.2021.3133882

[6] Fu, X., Fortino, G., Pace, P., Aloi, G., & Li, W. (2020). Environment-fusion multipath routing protocol for wireless sensor networks. Information Fusion, 53, 4-19. https://dx.doi.org/10.1109/ACCESS.2020.3133882

[7] Bhardwaj, A., & El-Ocla, H. (2020). Multipath routing protocol using genetic algorithm in mobile ad hoc networks. IEEE Access, 8, 177534-177548. https://dx.doi.org/10.1109/ACCESS.2020.3027043

[8] Džubur, A. H., Čaušević, S., Memić, B., Begović, M., Avdagić-Golub, E., & Čolaković, A. (2024). Optimization model proposal for traffic differentiation in wireless sensor networks. Computers, Materials & Continua, 81(1). https://dx.doi.org/10.32604/cmc.2024.055386

[9] Luo, J., Chen, Y., Wu, M., & Yang, Y. (2021). A survey of routing protocols for underwater wireless sensor networks. IEEE Communications Surveys & Tutorials, 23(1), 137-160. https://dx.doi.org/10.1109/COMST.2021.3048190

[10] Rani, S., Ahmed, S. H., & Rastogi, R. (2020). Dynamic clustering approach based on wireless sensor networks genetic algorithm for IoT applications. Wireless Networks, 26(4), 2307-2316. https://dx.doi.org/10.1007/s11276-019-02083-7

[11] Zhao, D., Li, Y., Zeng, Y., Wang, J., & Zhang, Q. (2022). Spiking capsnet: A spiking neural network with a biologically plausible routing rule between capsules. Information Sciences, 610, 1-13. https://dx.doi.org/10.1016/j.ins.2022.07.152

[12] Khudayer, B. H., Anbar, M., Hanshi, S. M., & Wan, T. C. (2020). Efficient route discovery and link failure detection mechanisms for source routing protocol in mobile ad-hoc networks. IEEE Access, 8, 24019-24032. https://dx.doi.org/10.1109/ACCESS.2020.2970279

[13] Shyur, H., & Shih, H. (2024). Resolving rank reversal in TOPSIS: a comprehensive analysis of distance metrics and normalization methods.

Informatica, 1-22. https://dx.doi.org/10.15388/24-INFOR576

[14] Zhou, X., Yang, X., Ma, J., Kevin, I., & Wang, K. (2021). Energy-efficient smart routing based on link correlation mining for wireless edge computing in IoT. IEEE Internet of Things Journal, 9(16), 14988-14997. https://dx.doi.org/10.1109/JIOT.2021.3077937

[15] Lakew, D. S., Sa'ad, U., Dao, N. N., Na, W., & Cho, S. (2020). Routing in flying ad hoc networks: A comprehensive survey. IEEE Communications Surveys & Tutorials, 22(2), 1071-1120. https://dx.doi.org/10.1109/COMST.2020.2982452

[16] Hong, L., Guo, H., Liu, J., & Zhang, Y. (2020). Toward swarm coordination: Topology-aware inter-UAV routing optimization. IEEE Transactions on Vehicular Technology, 69(9), 10177-10187. https://dx.doi.org/10.1109/TVT.2020.3003356

[17] Kilčiauskas, A., Bendoraitis, A., & Sakalauskas, E. (2024). Confidential transaction balance verification by the net using non-interactive zero-knowledge proofs. Informatica, 35(3), 601-616. https://doi:10.15388/24-INFOR564

[18] Khan, I. U., Qureshi, I. M., Aziz, M. A., Cheema, T. A., & Shah, S. B. H. (2020). Smart IoT control-based nature inspired energy efficient routing protocol for flying ad hoc network (FANET). IEEE Access, 8, 56371-56378. https://dx.doi.org/10.1109/ACCESS.2020.2981531

[19] Zis, T. P., Psaraftis, H. N., & Ding, L. (2020). Ship weather routing: A taxonomy and survey. Ocean Engineering, 213, 107697. https://dx.doi.org/10.1016/j.oceaneng.2020.107697

[20] Daanoune, I., Abdennaceur, B., & Ballouk, A. (2021). A comprehensive survey on LEACH-based clustering routing protocols in Wireless Sensor Networks. Ad Hoc Networks, 114, 102409. https://dx.doi.org/10.1016/j.adhoc.2021.102409

[21] Gao, H., Liu, C., Li, Y., & Yang, X. (2020). V2VR: reliable hybrid-network-oriented V2V data transmission and routing considering RSUs and connectivity probability. IEEE Transactions on Intelligent Transportation Systems, 22(6), 3533-3546. https://dx.doi.org/10.1109/TITS.2020.2983835

[22] Deebak, B. D., & Al-Turjman, F. (2020). A hybrid secure routing and monitoring mechanism in IoT-based wireless sensor networks. Ad Hoc Networks, 97, 102022. https://dx.doi.org/10.1016/j.adhoc.2020.102022

[23] Chen, X., Tang, J., & Lao, S. (2020). Review of unmanned aerial vehicle swarm communication architectures and routing protocols. Applied Sciences, 10(10), 3661. https://dx.doi.org/10.3390/app10103661

[24] Zhang, K., He, F., Zhang, Z., Lin, X., & Li, M. (2020). Multi-vehicle routing problems with soft time windows: A multi-agent reinforcement learning approach. Transportation Research Part C: Emerging Technologies, 121, 102861. https://dx.doi.org/10.1016/j.trc.2020.102861

[25] Kim, J., Jang, S., Park, E., & Choi, S. (2020). Text classification using capsules. Neurocomputing, 376, 214-221. https://dx.doi.org/10.1016/j.neucom.2020.10.033

[26] Wu, H., Alay, Ö., Brunstrom, A., Ferlin, S., & Caso, G. (2020). Peekaboo: Learning-based multipath scheduling for dynamic heterogeneous environments. IEEE Journal on Selected Areas in Communications, 38(10), 2295-2310. https://dx.doi.org/10.1109/JSAC.2020.3000365

[27] Pessoa, A., Sadykov, R., Uchoa, E., & Vanderbeck, F. (2020). A generic exact solver for vehicle routing and related problems. Mathematical Programming, 183, 483-523. https://dx.doi.org/10.1007/s10107-020-01523-z

[28] Sumathi, J., & Velusamy, R. L. (2021). A review on distributed cluster based routing approaches in mobile wireless sensor networks. Journal of Ambient Intelligence and Humanized Computing, 12(1), 835-849. https://dx.doi.org/10.1007/s12652-020-02088-7

[29] L. Cheng, Y. Wang, F. Cheng, C. Liu, Z. M. Zhao, and Y. Wang. (2024). A deep reinforcement learning-based preemptive approach for cost-aware cloud job scheduling. IEEE Transactions on Sustainable Computing, 9(3), 422-432. https://dx.doi.org/10.1109/TSUSC.2024.3303898

[30] Prokhorenko, V., & Babar, M. A. (2024). Offloaded data processing energy efficiency evaluation. Informatica, 35(3), 649-669. https://dx.doi.org/10.15388/24-INFOR567