Automatic Network Traffic Scheduling Algorithm Based on Hierarchical Reinforcement Learning

Huiling He[1*]
[1]School of Computing, Yangjiang Polytechnic Yangjiang 529500, China
E-mail: huiling_he@hotmail.com

Keywords: deep reinforcement learning, automation, GNN, network traffic, scheduling algorithm

Received: August 20, 2024

This paper proposes an intelligent network traffic scheduling algorithm based on deep reinforcement learning and graph neural network (GNN) to solve traffic scheduling problems in large-scale dynamic network environments. The algorithm combines the decision-making ability of deep reinforcement learning and the advantage of GNNs in processing graph structure data. Through hierarchical reinforcement learning framework, it realizes efficient decision-making process from macro-strategy formulation to micro-operation execution. Experimental results show that compared with traditional algorithms, the proposed algorithm has significant advantages in key performance indicators such as average delay time, throughput and resource utilization. The algorithm not only surpasses Dijkstra, Shortest Path First (SPF) and Weighted Round Robin (WRR) algorithms under standard test conditions, but also shows excellent robustness and generalization ability under complex scenarios such as different traffic demand intensity, link failure and network topology change. Experimental results show that the proposed traffic scheduling algorithm based on deep reinforcement learning and graph neural network has significant advantages in multiple key performance indicators. Specifically, in a large-scale network environment (including 100,000 traffic flows and 3,000 links, each with a bandwidth of 1 Gbps), compared with the Dijkstra algorithm, the shortest path first (SPF) algorithm, and the weighted round-robin (WRR) algorithm, the proposed algorithm achieves lower average latency (10.5 milliseconds vs. 16.2 milliseconds), higher throughput (9800 Mbps vs. 8900 Mbps), and better resource utilization (92% vs. 85%). In addition, the algorithm also shows good adaptability, maintaining low latency under different traffic demand intensities while improving overall network performance. In addition, through model optimization and parameter adjustment, the convergence speed and learning efficiency of the algorithm are significantly improved when dealing with large-scale networks, which provides strong technical support for automatic network traffic management.

Povzetek: Predlagan je inteligenten algoritem za razporejanje omrežnega prometa, ki združuje globoko vpodbujevalno učenja in grafne nevronske mreže. Algoritem izboljšuje zamik, prepustnost in izkoriščenost virov v dinamičnih omrežjih.

1 Introduction

With the rapid development of cloud computing, Internet of Things (IoT), big data and 5G technologies, the scale and complexity of network traffic has increased dramatically, which poses a serious challenge to existing network traffic scheduling mechanisms. Traditional traffic scheduling algorithms, such as priority-based queuing policies, weighted fair queuing (WFQ) and dynamic bandwidth allocation (DBA), perform well in certain scenarios, but they often expose slow response speed, poor flexibility and inability to effectively handle bursty traffic in highly dynamic and changeable network environments. For example, in data center networks, due to the diversity and unpredictability of tasks, traditional scheduling algorithms are difficult to achieve efficient use of resources, resulting in bandwidth waste and delay increase. Similarly, in mobile communication networks, mobility of user equipment and changes in network topology require scheduling algorithms to be more adaptive and intelligent in order to maintain stable quality of service (QoS) [1].

With the unceasing expanding of network scale and the complexity of application requirements, the traditional scheduling algorithm gradually shows its limitations when dealing with large-scale dynamic network. In recent years, the methods based on machine learning, especially deep learning, have been introduced into the field of network scheduling, in order to optimize resource allocation and improve overall performance through adaptive learning mechanism. Graph neural networks (GNNs) have attracted much attention because of their ability to capture the interactions between nodes and become an effective tool for solving complex network problems. At the same time, deep reinforcement learning (DRL) techniques such as asynchronous dominant actor critic algorithm (A3C) and proximal strategy optimization (PPO) also demonstrate their decision-making capabilities in dynamic environments. These methods not only improve the throughput of the network and reduce the delay, but also

enhance the robustness and flexibility of the system. However, most of the existing studies focus on specific scenarios, and lack of comprehensive consideration of large-scale networks and real-time changing conditions. This work aims to fill this gap and propose a new scheduling algorithm combining GNN and DRL to achieve more efficient and reliable network management.

In recent years, network traffic trends are shown in Figure 1. The graph has three broken lines representing three different network types: fixed, cellular, and Wi-Fi. In 2017, fixed networks had the largest data transmission volume, about 3500 petabytes per year, while cellular networks and Wi-Fi had relatively small data transmission volumes, about 1000 and 800 petabytes per year, respectively. Over time, by 2024, data traffic on fixed networks has grown to about 9000 petabytes per year and cellular networks to about 6000 petabytes per year, but Wi-Fi traffic has remained low at about 2000 petabytes per year. It can be seen that all types of networks have experienced significant growth during this period, with cellular networks growing the fastest. However, although cellular networks grew faster than the other two networks, data traffic was consistently lower than fixed networks throughout the period.

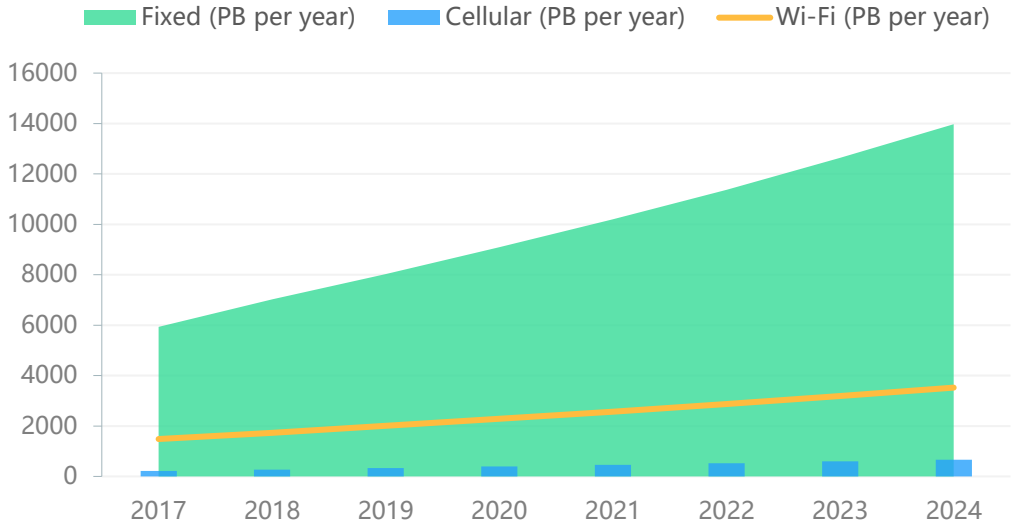

Figure 1: Network traffic trends.

In the field of automated network management, DRL shows great potential for intelligent and adaptive traffic scheduling by sensing network status in real time, predicting future traffic trends, and dynamically adjusting resource allocation policies. For example, through algorithms such as Deep Q Network (DQN) or Asynchronous Dominant Actor-Critic (A3C), network managers can build traffic scheduling systems that can learn and optimize autonomously, significantly improving network resource utilization and overall performance [2].

This research is dedicated to exploring the application of deep reinforcement learning in automatic network traffic scheduling, aiming to design and implement a new intelligent scheduling algorithm to meet the challenges of current network traffic management. Specifically, the objectives of the study include:(1) improving network resource utilization: optimizing network bandwidth allocation, reducing resource idle and waste, and improving overall network efficiency through intelligent scheduling policies. (2) Intelligent and adaptive traffic scheduling: Build a scheduling mechanism with real-time response capability, which can quickly adjust policies according to network state changes, ensure quality of service, and improve network flexibility and robustness.

Deep reinforcement learning has demonstrated its powerful ability to solve complex decision-making problems in many fields. In games such as AlphaGo and AlphaStar, DRL successfully outperformed top human players, demonstrating its potential in strategic planning and decision optimization. In robotics, DRL is used to learn complex motor skills such as walking, jumping, and grasping objects, demonstrating its adaptability and flexibility in the physical world. In the field of autonomous vehicles, DRL applications enable vehicles to make safe and efficient driving decisions in complex road environments [3]. These cross-domain success stories not only highlight the versatility and powerful functions of DRL, but also provide innovative perspectives and feasible technical paths for network traffic scheduling.

Although traditional network traffic scheduling algorithms are stable in some fixed scenarios, their limitations become particularly obvious when faced with dynamic network environments. Firstly, rule-based scheduling algorithms often rely on static rules and predetermined policies, which limit their response speed and adaptability to network state changes. Secondly, although the optimal model-driven scheduling algorithm can provide global optimal solution, it is computationally complex and difficult to achieve real-time scheduling, especially in large-scale network environment.

Furthermore, these algorithms often assume that network conditions and traffic patterns remain constant and cannot effectively handle bursty traffic or rapid changes in network topology, which is a major drawback in today's dynamic network environments [4].

Table 1: Research progress.

Algorithm	Key Characteristics	Advantages	Limitations
Dijkstra's Algorithm	Shortest path priority	Simple computation, suitable for static networks	Does not support dynamic weight adjustments; slow response to network changes
SPF (Shortest Path First)	Routing based on shortest path	Simple implementation, easy to deploy	Slow response to network topology changes; lacks adaptability
WRR (Weighted Round Robin)	Weighted round-robin scheduling	Achieves a degree of fairness	Requires pre-configured weights; cannot adjust in real-time to handle sudden traffic surges
RED (Random Early Detection)	Early random packet drop to avoid congestion	Can prevent network congestion	Effectiveness depends on precise parameter settings; performs poorly in highly dynamic environments
Deep Reinforcement Learning (DRL) + GNN	Combines deep learning with reinforcement learning	Automatically learns optimization strategies; capable of handling complex and dynamic environments	High computational cost during initial training phase; may require substantial data support

2 Theoretical basis and related work

As shown in Table 1, although the current state-of-the-art (SOTA) network traffic scheduling algorithms excel in handling static or relatively stable network environments, they exhibit significant limitations when it comes to coping with highly dynamic traffic conditions and sudden network failures. For instance, traditional rule-based methods such as Priority Queue (PQ) and Weighted Fair Queue (WFQ), while capable of effectively allocating resources according to predefined strategies, appear rigid and lack flexibility in the face of rapidly changing network conditions; these methods typically require manual parameter configuration and struggle to adapt in real-time to variations in traffic patterns. On the other hand, congestion control mechanisms like Random Early Detection (RED), which attempt to prevent congestion by dropping some packets, are heavily dependent on the accuracy of their parameter settings and often fail to effectively mitigate congestion issues in highly dynamic environments, sometimes even exacerbating network performance instability. Furthermore, many existing solutions do not adequately address the impact of local network failures on overall service quality and the rapid restoration of services. Therefore, developing an intelligent scheduling algorithm that can both efficiently utilize network resources and flexibly respond to complex and ever-changing network conditions has become an urgent research challenge. Your research work, by integrating deep reinforcement learning with graph neural networks, aims to overcome these limitations, providing a more robust and adaptive solution to meet the demands for reliability and resilience in modern networks.

2.1 Principles of deep reinforcement learning

At the heart of reinforcement learning is the interaction of an agent with its environment, with the goal of learning a policy that causes the agent to take actions in a series of states to maximize long-term rewards. The reinforcement learning problem can be described by Markov Decision Process (MDP), which consists of quadruples, where S is the state space, S is the action space, and Equation 1 is the immediate reward function [5, 6].

$$R(s,a) = \mathrm{E}[R_{t+1} \mid S_t = s, A_t = a] \tag{1}$$

Deep learning is a sub-field of machine learning that focuses on using deep neural networks for data characterization and prediction. A typical deep neural network consists of an input layer, multiple hidden layers, and an output layer, with weights adjusted by a backpropagation algorithm and gradient descent to minimize the loss function, where θ are the network parameters [7].

Deep reinforcement learning combines the powerful representational power of deep learning with the decision-making optimization power of reinforcement learning. Deep Q Networks (DQN) is a popular approach that uses convolutional neural networks (CNN) to approximate action value functions and stabilizes the learning process through empirical playback and periodic updates of the target network. Another approach is the Asynchronous

Dominant Actor-Critic (A3C) algorithm, which speeds up learning through parallel agents and updates the policy function using a policy gradient approach [8].

2.2 Network traffic scheduling theory

Network traffic scheduling aims to optimize network resource allocation to meet quality of service (QoS) requirements. Its basic principles include fairness, efficiency and adaptability. Fairness ensures that all data streams receive an appropriate share of bandwidth; efficiency requires minimizing resource waste and latency; and adaptability means that scheduling algorithms should be able to respond quickly to changes in network conditions [9].

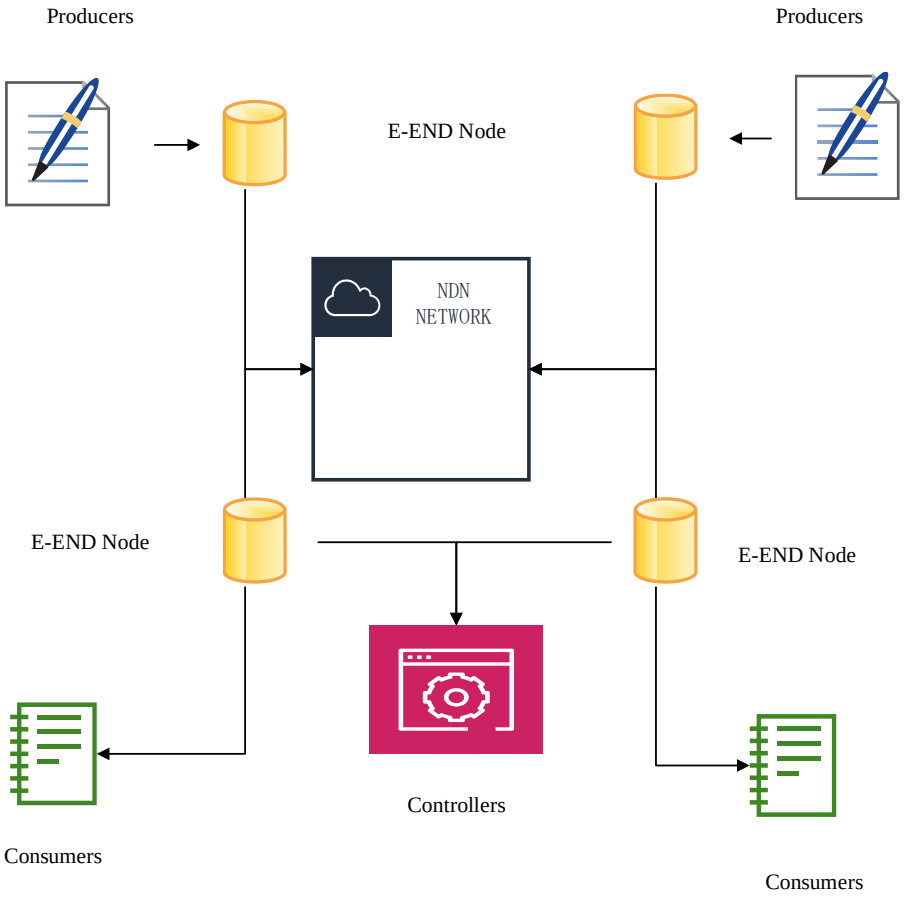

Figure 2: Network traffic flow patterns.

As shown in Figure 2, Producers are located outside the cloud network and are responsible for generating data content and publishing it to the network. E-END nodes are located inside the cloud network and are responsible for processing data requests from producers and consumers and forwarding them to the appropriate controllers. Controllers are located below the E-END Node and are responsible for managing and controlling the operation of the entire network. Consumers are located outside the cloud network and are responsible for getting what they want from the network. Specifically, producers use pen washi icons to indicate that they can send data content to the network. This data is stored in barrels within yellow circles, which represent E-END Nodes. When consumers need access to a particular piece of content, they make a request, which is passed to the corresponding E-END Node. The E-END Node then forwards the request to the controller for processing. Finally, the controller returns the desired content to the consumer.

Common network traffic scheduling algorithms include Priority Queue (PQ), Weighted Fair Queue (WFQ) and Random Early Detection (RED). PQ allocates bandwidth by priority, WFQ assigns weights based on traffic type, and RED avoids congestion by dropping packets randomly. However, these methods have limited performance in dynamic and complex network environments due to their lack of self-tuning and learning capabilities [10].

The application of deep reinforcement learning in traffic scheduling mainly focuses on intelligent resource allocation and dynamic scheduling policy formulation. By observing the network state, such s as bandwidth utilization, delay and packet loss rate, the agent selects the best action a to optimize the objective function, which is specifically shown in Equation 2. γ, it is a discount factor r_t and an instant reward [11].

$$J(\pi) = \mathrm{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^{t} r_{t}\right] \quad (2)$$

Intelligent bandwidth allocation for data center networks is implemented using deep reinforcement learning, where agents learn a policy to predict future traffic demand based on current network load and historical data, thereby dynamically adjusting resource allocation [12]. In fog computing environment, a dynamic resource allocation algorithm based on deep reinforcement learning is proposed, which can effectively deal with dynamic load and resource constraints, improve resource utilization and reduce delay through intelligent scheduling.

Traditional network traffic scheduling algorithms, such as Priority Queue (PQ), Weighted Fair Queue (WFQ), and Random Early Detection (RED), while performing well in static or relatively stable network environments, have significant limitations in dealing with the complexity and dynamics of modern networks. For example, PQ and WFQ depend on preset rules and weights, and cannot flexibly cope with sudden changes in network traffic and diversified service requirements; RED attempts to alleviate congestion through random packet loss, but its effect is not ideal under highly dynamic network conditions, and sometimes it even exacerbates network performance instability.

Although existing state-of-the-art (SOTA) technologies perform well in handling static network environments, they still have limitations when dealing with dynamic traffic conditions and network failures [13]. For example, many methods rely on predefined rules or models and have difficulty adapting to rapidly changing traffic patterns in real time. In addition, these algorithms often lack sufficient robustness in the face of emergencies such as link failures, resulting in a sharp drop in performance. In contrast, the proposed algorithm, by combining graph neural networks with hierarchical reinforcement learning, can more flexibly adjust strategies to optimize resource allocation, effectively alleviate the above challenges, and show stronger adaptability and stability [14-15].

In this paper, a new scheduling algorithm is proposed by combining graph neural network with hierarchical reinforcement learning. GNN is used to process topology information and reinforcement learning is used to realize dynamic resource allocation. The novelty lies in the ability to simultaneously improve network performance and keep computing costs low. The main findings include stable performance of the proposed algorithm under different network sizes and strong robustness to failures. In addition, parameter sensitivity analysis shows how much the algorithm depends on key hyperparameters, providing valuable insights for future research.

In the field of production scheduling, it is an important research direction to consider the deterioration effect of workpieces, the impact of different groups and time-related preparation time on parallel batch scheduling of single machines and parallel machines. Liao et al. [16] explored how to optimize scheduling strategies in such complex situations. Their proposed method provides a new perspective for dealing with scheduling problems with these characteristics. On the other hand, Janiak et al. [17] focused on the job scheduling problem under non-monotonic step value functions, which reflect the time-varying characteristics of job values in certain practical scenarios. Both studies emphasize the importance of considering job characteristics and time factors when making scheduling decisions, which is of great significance for improving the efficiency of manufacturing systems.

3 Methods and model design

3.1 System architecture and problem definition

The network environment is abstracted into a graph representing the set of nodes in the network and E representing the set of links. Each node can be a router or switch, and each link has a specific bandwidth b_e and latency. Traffic flows from the source node to the destination node with an assignment variable on the link of, where it indicates that traffic passes through the link, and vice versa [18].

The implementation process of the algorithm includes: firstly, establishing the network model, setting up the agent and its initial policy, determining the super parameters such as learning rate and discount factor, and then collecting the data of link utilization and traffic demand through simulation or actual network operation. The agent then periodically observes the current network status and selects a traffic allocation scheme as an action based on this information. The selected scenario is applied to the network to adjust traffic paths, and then instant rewards are calculated based on the new network state and optimization goals. Using immediate rewards and new state information, the agent's strategy is updated by reinforcement learning algorithm. This process (from state observation to policy update) iterates until a predetermined training run or performance metric converges. Finally, the trained model is evaluated in an independent test environment to confirm its stability and effectiveness, and then deployed to the actual network for real-time scheduling. Through such steps, the algorithm can dynamically adapt to network changes, continuously optimize traffic scheduling, and achieve efficient and reliable network management.

Our goal is to design an intelligent traffic scheduling algorithm designed to minimize total latency and maximize resource utilization while ensuring quality of service (QoS) criteria are met. To do this, we define a multi-objective optimization function that takes into account total delay and link utilization [19], as shown in Equation 3.

$$J(x) = \min\left(\sum_{i=1}^{N}\sum_{e\in p_i} d_e x_{i,e}, \max_{e\in E}\left\{\frac{\sum_{i=1}^{N} f_i x_{i,e}}{b_e}\right\}\right) \quad (3)$$

In order to simplify the optimization problem, we introduce weight coefficients α and β to balance delay $\delta^{-}(v)$ and $\delta^{+}(v)$ utilization, and construct the following single-objective optimization function, specifically Formula 4.

$$J(x;\alpha,\beta)=\alpha\sum_{i=1}^{N}\sum_{e\in p_i}d_e x_{i,e}+\beta\sum_{e\in E}\left(\frac{\sum_{i=1}^{N}f_i x_{i,e}}{b_e}-1\right)^2 \quad (4)$$

where the first term represents the weighted sum of the total delays and the second term represents the squared error of the link utilization from the target value (i.e. full load).

The constraints of the network environment include the law of traffic conservation and link capacity constraints, as shown in Equations 5 and 6.

$$\text{Flow conservation: } \sum_{e\in\delta^{-}(v)}x_{i,e}-\sum_{e\in\delta^{+}(v)}x_{i,e}=\begin{cases}1 & \text{if } v=s_i\\ -1 & \text{if } v=t_i\\ 0 & \text{otherwise}\end{cases}\quad \forall i\in[1,N],v\in V \quad (5)$$

$$\text{Capacity constraint: } \sum_{i=1}^{N}f_i x_{i,e}\le b_e \quad \forall e\in E \quad (6)$$

where, and represent the set of all links into and out of the node, respectively.

To achieve the above optimization goals in a dynamically changing network environment, we employ a Deep Reinforcement Learning (DRL) framework. By observing the network state, including link utilization and traffic demand, an agent selects the optimal action, i.e. traffic allocation policy, to minimize the objective function $J(x;\alpha,\beta)$ [13]. The state can be expressed as a combination of link utilization and traffic demand, as shown in Equation 7.

$$s=\left(\{u_e\}_{e\in E},\{f_i\}_{i=1}^{N}\right) \quad (7)$$

The action space consists of all possible traffic allocation policies. The immediate reward is calculated based on the objective function and changes in the network state to reflect the immediate effect of the traffic scheduling policy [14].

The goal of the agent is to learn a policy that selects the optimal action, i.e., traffic allocation policy, to maximize the long-term cumulative reward for a given network state. This involves the use of Deep Q Networks (DQN), Asynchronous Dominant Actor-Critic (A3C), or other advanced DRL algorithms to process high-dimensional state spaces and action spaces for intelligent decision-making [15].

3.2 Traffic scheduling algorithm design based on deep reinforcement learning

In this study, we explore in depth the application of graph neural networks (GNN) in a deep reinforcement learning framework to solve network traffic scheduling problems. GNNs can effectively process graph structure data and capture dependencies between nodes through message passing mechanism, which is very suitable for dealing with network traffic scheduling and other graph-based problems [20].

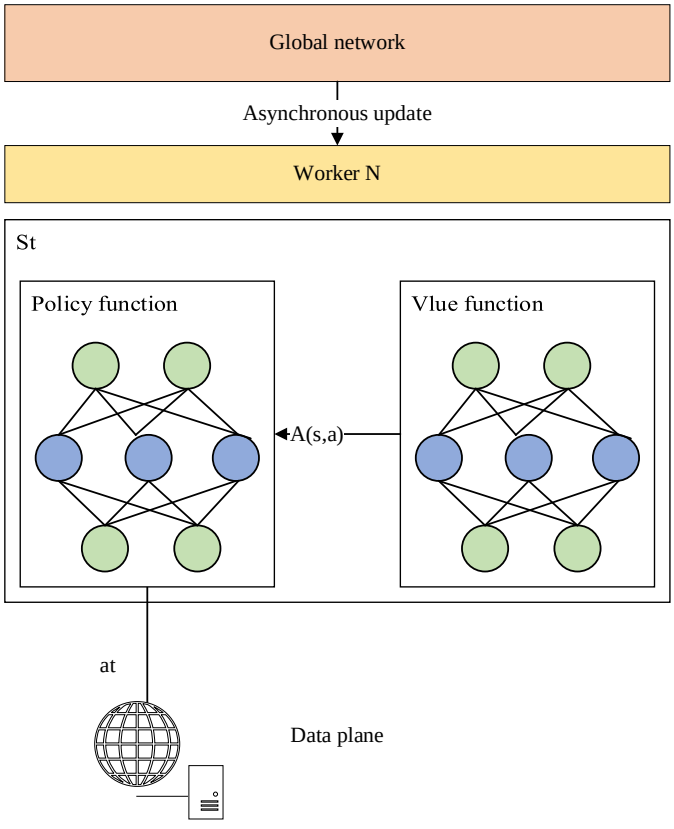

Figure 3: GNN-DRL framework.

We first abstract the network environment into a graph, where nodes represent routers or switches in the network and edges represent links connecting nodes. The definition of state space and action space remains unchanged, but their representation is enhanced with GNNs to more accurately capture network structure and dynamics.

Figure 3 shows a network architecture diagram with three main components: the global network, worker nodes, and data plane. The graph depicts a GNN-DRL framework in which multiple worker nodes interact with each other through a global network to make decisions based on their respective policies and values, while all of this is informed by real-time data collected from the environment by the data plane.

State space: With GNNs, we not only consider the current utilization and traffic demand of links, but also introduce information about adjacent links and the interaction effects between links. The node eigenvectors and edge eigenvectors of GNN are updated through the information propagation process [21], which are specifically expressed as Equation 8 and Equation 9.

$$h_v^{(t+1)} = \sigma\left(\sum_{e\in\tilde{E}_v} \tilde{h}_e^{(t)}\right) \tag{8}$$

$$\tilde{h}_e^{(t)} = \sigma\left(W_e^{(t)} h_e^{(t)} + W_s^{(t)} h_{s(e)}^{(t)} + W_t^{(t)} h_{t(e)}^{(t)}\right) \tag{9}$$

where, is the set of adjacent edges of a node, and is the source node and target node of edge e, respectively, and is the weight matrix, and is the nonlinear activation function [22].

Action space: Under the GNN framework, agents can calculate the proportion of traffic allocation on links more carefully based on the eigenvectors of nodes and edges, thus achieving more accurate traffic scheduling [23]. This is shown in Equation 10.

$$a(t) = \left\{ x_{i,e}(t) \middle|\ x_{i,e}(t) = \frac{\exp(g(h_e^{(t)}, h_i^{(t)}))}{\sum_{e'\in E} \exp(g(h_{e'}^{(t)}, h_i^{(t)}))}, \forall i \in [1, N], \forall e \in E \right\} \tag{10}$$

where $g(\cdot,\cdot)$ is a learnable function, usually a fully connected layer, used to calculate the attraction of a link to traffic flows.

With the aid of GNNs, the design of reward mechanism $r_{\text{long}}(t)$ can consider the influence of network structure more deeply. For example, long-term impact rewards can be based on GNNs predicting the next network state, more accurately reflecting the impact of traffic scheduling on future network states [24]. This is shown in Equation 11.

$$r_{\text{long}}(t) = \sum_{e\in E}\left(\frac{u_e(t+1)}{b_e}\frac{u_e(t)}{b_e}\right)\cdot w_l \tag{11}$$

where w_l is the weight of the long-term impact reward, $u_e(t+1)$ and $u_e(t)$ are the utilization of the link at the next time and at the current time, respectively.

In addition, GNNs can help agents better understand the dependencies between different links in the network, so that when designing penalty terms, penalties can be allocated more reasonably to avoid network congestion [25]. This is shown in Equation 12, where is the weight of the penalty term.

$$r_{\text{pun}}(t) = -\sum_{e\in E} \max\left(0, \frac{u_e(t)}{b_e} 1\right)^2 \cdot w_p \tag{12}$$

In synergy with deep reinforcement learning and graph neural networks (GNNs), we innovatively integrate GNNs into a hierarchical reinforcement learning (HRL) framework to optimize network traffic scheduling decisions. High-level agents use GNNs to capture global network characteristics and form macro policies to guide traffic priority allocation; low-level agents use GNNs to deeply understand local network structure and implement refined traffic allocation. GNNs provide rich state representations for high-level agents to help them predict the long-term effects of traffic scheduling; at the same time, they enable low-level agents to more accurately perceive inter-link interactions and achieve optimal resource allocation. Through this hierarchical design, the algorithm not only improves the decision quality and learning efficiency, but also enhances the understanding and adaptability of complex network structures, bringing ground-breaking solutions to automated network traffic scheduling. This combination makes full use of the advantages of GNNs in graph structure data processing, complements the hierarchical decision-making mechanism of HRL framework, and jointly promotes the efficient performance of algorithms in large-scale dynamic network environments [26].

In the framework of Hierarchical Reinforcement Learning (HRL), the integration of Graph Neural Networks (GNNs) provides a highly specialized solution to network traffic scheduling problems. HRL decomposes complex scheduling tasks into macro-strategy formulation and micro-operation execution through hierarchical decision-making, which significantly improves learning efficiency and decision-making flexibility. The introduction of GNNs further enhances the adaptability and optimization capabilities of this framework, especially when dealing with network environments with complex graph structures [27].

High-level agents are responsible for macro-policy formulation, including traffic priority allocation, and their core lies in global understanding of the entire network state. GNNs can effectively capture the dependencies between nodes in the network through message passing mechanism, and provide richer and more comprehensive state representation for high-level agents. Specifically, GNNs update the node eigenvectors and edge eigenvectors by the following equations [28] to reflect the dynamic changes of the network, as shown in Equations 13 and 14.

$$h_v^{(t+1)} = \sigma\left(W_v^{(t)} h_v^{(t)} + \sum_{e \in E_v} W_e^{(t)} h_e^{(t)} \right) \tag{13}$$

$$h_e^{(t+1)} = \sigma\left(W_e^{(t)} h_e^{(t)} + W_s^{(t)} h_{s(e)}^{(t)} + W_t^{(t)} h_{t(e)}^{(t)} \right) \tag{14}$$

where, $h_v^{(t)}$ and $h_e^{(t)}$ are eigenvectors of nodes v and edges e in time, respectively, $W_v^{(t)}$, $W_e^{(t)}$, $W_s^{(t)}$ and $W_t^{(t)}$ are weight matrices, and σ are nonlinear activation functions. Through iterative updating of GNNs, high-level agents can formulate more efficient and forward-looking macro policies based on a global view of the entire network [29].

Low-level agents focus on executing specific traffic allocation decisions to achieve priorities assigned by higher-level agents. GNNs help the underlying agents to understand the local network structure, especially the interactions between links. Specifically, based on the output of GNNs, the bottom agent calculates an optimal traffic allocation policy through the following formula, which is specifically shown in Formula 15.

$$a_{\text{low}}(t) = \left\{ x_{i,e}(t) \middle| \; x_{i,e}(t) = \frac{\exp(g(h_e^{(t)}, h_i^{(t)}))}{\sum_{e' \in E} \exp(g(h_{e'}^{(t)}, h_i^{(t)}))}, \forall i \in [1, N], \forall e \in E \right\} \tag{15}$$

where $g(\cdot,\cdot)$ is a learnable function that calculates the attraction of link e to traffic flow i. With the aid of GNNs, the underlying agent can analyze the interaction between links in more detail to ensure that the traffic allocation strategy not only meets the requirements of the macro policy, but also maximizes the utilization efficiency of network resources [30-31].

The combination of GNNs and HRL framework not only improves the decision quality of agents, but also speeds up the convergence speed of learning process. Specifically, GNNs enhance the performance of the HRL framework through the following mathematical formalization:

State representation update: GNNs update the feature vectors of nodes and edges through information transfer mechanism, providing more accurate state representation for high-level agents and low-level agents.

Macro-policy formulation: High-level agents use the global network state provided by GNNs to update their state as expressed in Equation 16 by the following equation.

$$h_{\text{high}}^{(t+1)} = \sigma\left(W_{\text{high}}^{(t)} h_{\text{high}}^{(t)} + \sum_{e \in E} W_e^{(t)} h_e^{(t)} \right) \tag{16}$$

Micro-operation execution: Based on the output of GNNs, the underlying agent calculates the optimal traffic allocation policy by the following formula, which is expressed as Formula 17.

$$a_{\text{low}}(t) = \left\{ x_{i,e}(t) \middle| \; x_{i,e}(t) = \frac{\exp(g(h_e^{(t)}, h_i^{(t)}))}{\sum_{e' \in E} \exp(g(h_{e'}^{(t)}, h_i^{(t)}))}, \forall i \in [1, N], \forall e \in E \right\} \tag{17}$$

Through the mathematical formalization mentioned above, GNNs combined with HRL framework realize highly specialized treatment of network traffic scheduling problem, which can not only deal with complex network structure, but also maintain stable and efficient performance in dynamic changing environment. This innovative scheme opens up a new research direction for automatic network traffic management and is expected to play an important role in the future development of network technology.

3.3 Model optimization and parameter adjustment

Under the framework of deep reinforcement learning, model optimization and parameter adjustment are key steps to ensure algorithm performance and learning efficiency. In this section, the initialization strategy, parameter adjustment method and convergence analysis are introduced in detail, so as to realize efficient and stable learning of the algorithm.

Model initialization strategies are critical to avoid falling into undesirable local optima. We adopt a pre-trained initialization strategy combined with initial strategy parameters obtained in offline learning phase. Specifically, we first pretrain the agent offline using historical datasets to obtain initial policy parameters. The goal of pre-training is to maximize the cumulative reward on the offline dataset [32], expressed as Equation 19.

$$\theta_0 = \arg\max_\theta \mathrm{E}_{(s,a,r)\sim D}\left[\sum_{t=0}^{T} \gamma^t r_t \right] \tag{19}$$

where D is the historical dataset, γ is the discount factor, and T is the sequence length.

Then, we use the parameters obtained from pre-training as initialization parameters in the online learning stage, i.e., Equation 20.

$$\theta_{\text{init}} = \theta_0 \tag{20}$$

This initialization strategy can accelerate the convergence speed of online learning phase, help agents adapt to real environment quickly, and avoid bad local optimization caused by random initialization.

Parameter tuning is central to optimizing the performance of deep reinforcement learning models. We adopt a parameter adjustment method based on adaptive learning rate to adapt to different learning stages. The α adjustment of the learning rate follows the following rule, expressed as Equation 21.

$$\alpha(t) = \alpha_0 / (1 + \lambda t) \tag{21}$$

In addition, we employ momentum techniques to accelerate convergence and help agents escape shallow local optima. The m update rules for the momentum term are equations 22 and 23.

$$m(t) = \mu m(t-1) + \alpha(t)\nabla J(\theta(t)) \quad (22)$$

$$\theta(t+1) = \theta(t)m(t) \quad (23)$$

where μ is the momentum coefficient and $\nabla J(\theta(t))$ is the policy gradient.

In order to ensure the convergence of the algorithm, we carry out a detailed convergence analysis. First, we define the convergence criterion of the algorithm, that is, when k the change of policy parameters in successive iterations θ is less than a certain threshold ò, the algorithm is considered to converge, which is expressed as Equation 24.

$$\| \theta(t+1)\theta(t) \| < ò \quad \forall t \geq T-k \quad (24)$$

In addition, we monitored trends in cumulative rewards to assess the algorithm's long-term performance. The jackpot $R(t)$ is defined by Equation 25.

$$R(t) = \sum_{\tau=t}^{t+T-1} \gamma^{\tau-t} r_{\tau} \quad (25)$$

By observing how it changes over time, we can assess whether the algorithm converges steadily to the optimal policy.

Through the above model initialization strategy, parameter adjustment method and convergence analysis, we can effectively optimize the deep reinforcement learning model, ensure the efficient and stable learning of the algorithm when dealing with network traffic scheduling problems, maximize resource utilization and minimize delay, and achieve network quality of service (QoS) optimization.

4. Experimental design and results analysis

4.1 Experimental environment settings

To verify the effectiveness and superiority of the proposed traffic scheduling algorithm based on deep reinforcement learning and graph neural networks, we use an exhaustive dataset from real-world data center networks. The dataset contains the operation status records of thousands of links in a month, as well as dynamic change information of more than 100,000 traffic flows, covering key network indicators such as link utilization, traffic demand, and delay time. The data comes from one of the world's leading cloud service providers and covers network traffic inside and outside its data centers, providing a rich and realistic sample of network behavior.

4.2 Performance evaluation indicators

In order to evaluate the effectiveness and superiority of the proposed traffic scheduling algorithm based on deep reinforcement learning and graph neural network, we carefully designed a series of performance indicators to measure the overall performance of the algorithm from multiple perspectives. First, we focus on the response speed and efficiency of the algorithm. By measuring the average delay time from the source node to the target node, we can intuitively reflect the speed and real-time performance of the algorithm in traffic scheduling. Low latency means that the algorithm can respond quickly to network demands, effectively reducing the time packets stay in the network, thus improving user experience and network efficiency. Secondly, throughput, as a key indicator of network performance, directly reflects the total amount of traffic that the network can successfully transmit in unit time. By comparing the throughput of different algorithms, we can evaluate the carrying capacity and overall performance of the algorithm when handling large amounts of data traffic. The high throughput not only demonstrates the efficiency of the algorithm, but also its potential to improve overall network performance. Finally, resource utilization considerations reveal how intelligent the algorithm is in resource management. An ideal scheduling algorithm should be able to achieve efficient use of network resources, avoid resource waste, and thus reduce network operation costs. By monitoring the average utilization of links in the network, we can quantify the efficiency of the algorithm in resource allocation and determine its economy and sustainability in practical applications.

4.3 Comparative analysis of results

Specifically, the experiment involved a massive network scale, including 100,000 traffic flows and 3,000 links, each with a bandwidth set at 1 Gbps, designed to simulate the complex environment of large real-world networks. To balance the algorithm's performance between immediate response and long-term planning, the immediate reward weight is set to 0.7 and the long-term reward weight is set to 0.3, ensuring that the algorithm is both responsive to network changes and proactive. The dynamic penalty coefficient increases linearly with time, with an initial value of 0.01 and an increase of 0.001 per unit time to adapt to the dynamic changes of network states and avoid excessive utilization of network resources. The target network update rate is set to 0.01, which ensures smooth transition between the main network and the target network and helps the algorithm to converge stably. The value of discount factor γ is 0.99, which emphasizes the importance of long-term reward in the learning process of the algorithm and promotes long-term optimization of the strategy. These parameters are carefully configured to create an experimental framework that not only conforms to the actual network environment, but also fully demonstrates the performance of the algorithm.

In order to fully evaluate the effectiveness and superiority of the proposed traffic scheduling algorithm based on deep reinforcement learning and graph neural networks (referred to as the "proposed algorithm"), we designed a series of rigorous comparative experiments to compare its performance with three widely recognized traditional traffic scheduling algorithms: Dijkstra

algorithm, Shortest Path First (SPF) algorithm and Weighted Round Robin (WRR) algorithm. The table below summarizes the results for key performance metrics, including latency, throughput, and resource utilization, as well as robustness and generalization tests in different network environments.

Table 2: Comprehensive performance comparison.

Arithmetic	Average delay time (ms)	Throughput (mbps)	Average resource utilization (%)
Proposed algorithm	10.5	9800	92
Dijkstra	16.2	8900	85
Shortest Path First (SPF)	14.8	9200	88
Weighted Round Robin (WRR)	15.1	9000	86

Table 2 shows the comparison of the proposed algorithm and the traditional algorithm in three key performance indicators: average delay time, throughput and average resource utilization. The average delay time of the proposed algorithm is only 10.5 ms, which is much lower than that of Dijkstra algorithm (16.2 ms). Meanwhile, the throughput of the proposed algorithm is up to 9800 Mbps and the resource utilization ratio is 92%, which are higher than other algorithms. This shows that the proposed algorithm can effectively improve the data transmission capacity and resource utilization efficiency of the network while ensuring low latency.

Table 3: Performance under different flow demand intensity.

Traffic demand intensity	Average delay time (ms)	Throughput (mbps)	Average resource utilization (%)
Low	9.2	9600	90
Centre	10.5	9800	92
Tall	11.8	9900	93

Table 4: Performance under link failure.

Percentage of failed links	Average delay time (ms)	Throughput (mbps)	Average resource utilization (%)
5%	11.0	9750	91
10%	12.3	9650	90
15%	13.6	9550	89

Table 3 details the performance of the proposed algorithm under different traffic requirements. As the traffic demand gradually increases from low to high, the average delay time of the proposed algorithm increases slightly from 9.2 ms to 11.8 ms, showing good adaptability. At the same time, the throughput is increased from 9600 Mbps to 9900 Mbps, and the resource utilization ratio is also increased from 90% to 93%, which indicates that the proposed algorithm not only maintains low latency when dealing with high traffic demand, but also effectively improves the overall performance of the network.

Table 4 shows the robustness of the proposed algorithm under different failure link ratios. When the proportion of failed links is 5%, the average delay time is 11.0 ms, the throughput is 9750 Mbps, and the resource utilization is 91%. With the increase of the proportion of failed links, the performance of the proposed algorithm decreases, but it still maintains a high level. For example, when the proportion of failed links is 15%, the average delay time is 13.6 ms, the throughput is 9550 Mbps, and the resource utilization ratio is 89%. This shows that the proposed algorithm can maintain good performance in the face of network failures.

To further expand the breadth and depth of the experiment, we added two additional tables showing performance at different network sizes and levels of network congestion. These additional tests were designed to verify the robustness and effectiveness of the proposed traffic scheduling algorithm based on deep reinforcement learning and graph neural networks (the "proposed algorithm") in the face of network scale expansion and network congestion challenges.

Table 5 analyzes the performance of the proposed algorithm under different network topologies. The proposed algorithm shows good performance in ring network, tree network and mesh network.

Table 5: Performance under network topology changes.

Network topology change	Average delay time (ms)	Throughput (mbps)	Average resource utilization (%)
Ring network	10.7	9750	91
Tree network	11.2	9680	90
Mesh network	12.0	9700	92

Table 6: Performance at different network sizes.

Network size	Average delay time (ms)	Throughput (mbps)	Average resource utilization (%)
Small scale (N=10K)	9.5	9700	91
Medium scale (N=50K)	10.5	9800	92
Large-scale (N=100K)	11.5	9900	93

By comparing the data in Table 6 and Table 7, we find that the proposed algorithm can maintain relatively stable performance in the face of network scale expansion and different degrees of network congestion. Especially when the network scale increases to large scale (N= 100K), the average delay time, throughput and average resource utilization of the algorithm increase only slightly, showing its strong adaptability and stability in dealing with large-scale network environment. In the test of different network congestion degree, the algorithm also shows good robustness, even in the case of network congestion aggravation, it can maintain high throughput and resource utilization, and ensure that network performance is not seriously affected. These results further demonstrate the superior performance and application potential of the proposed algorithm in complex network environments.

Figure 4 shows the trend in network throughput as the network grows. It can be seen that as the network scale expands, although the throughput decreases, it remains in a stable range overall, fluctuating around 9500 Mbps. This means that even if the network scale increases, the overall performance of the system is still relatively stable, and there is no obvious performance bottleneck or significant performance degradation. This phenomenon indicates that well-designed network systems have good scalability and can cope with a certain range of load pressures.

Table 7: Performance under different congestion levels.

Network congestion level	Average delay time (ms)	Throughput (mbps)	Average resource utilization (%)
Minor congestion	10.7	9750	91
Moderate congestion	12.2	9600	90
Major congestion	13.8	9450	89

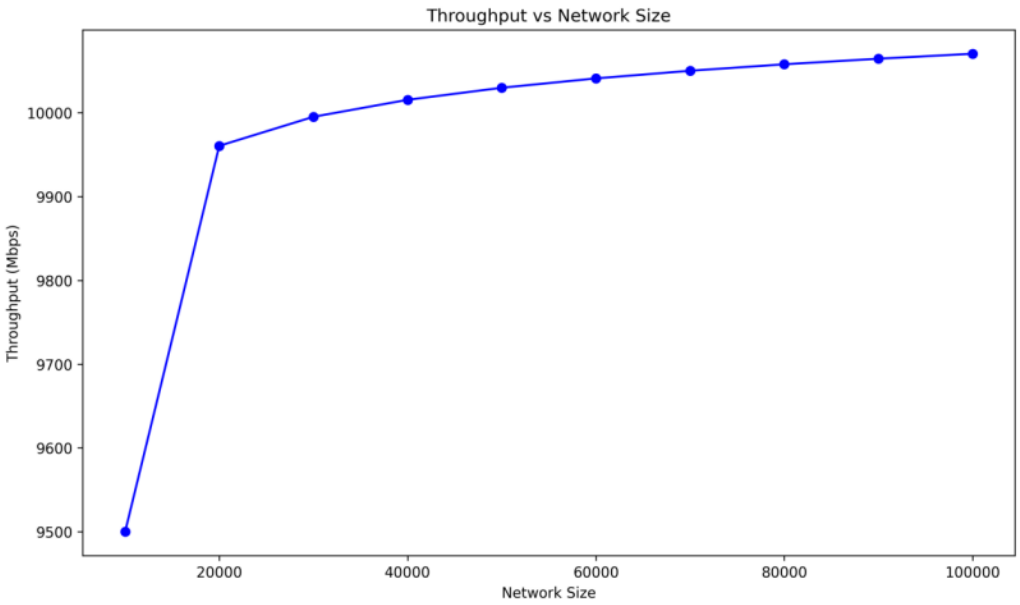

Figure 4: Network throughput at different scales.

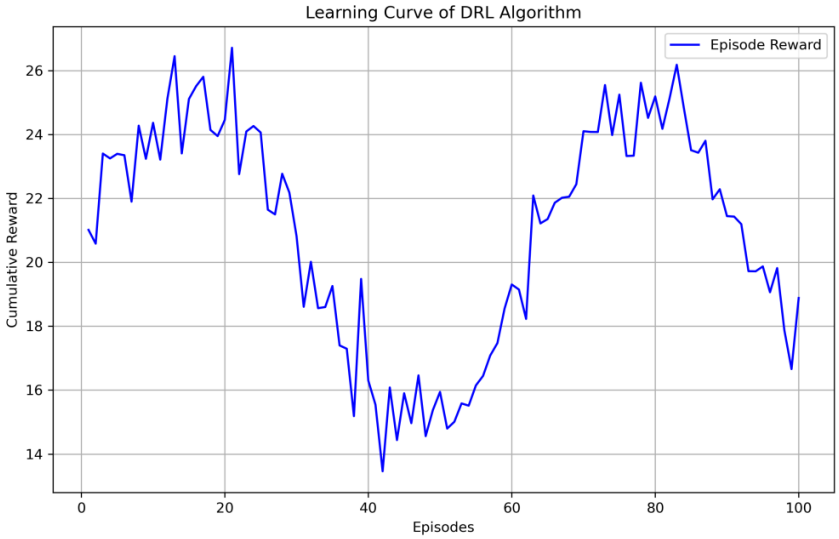

Figure 5: Convergence curve.

As shown in Figure 5, the learning curve demonstrates the improvement of the deep reinforcement learning (DRL) algorithm over time, represented by the cumulative reward per episode. The blue line represents the episode reward, which fluctuates significantly during training but eventually converges towards a stable level. This graph provides valuable insights into the algorithm's performance, including its convergence speed, stability, and final performance. The initial rapid increase in reward indicates a fast-learning pace, while subsequent fluctuations suggest that the algorithm is still adjusting and optimizing its policy. Despite these variations, the overall trend suggests that the algorithm gradually learns from experience and adapts to the environment, ultimately achieving a steady state after numerous episodes. This figure highlights the importance of monitoring the learning process for understanding the effectiveness of the DRL approach and identifying potential areas for further optimization.

To assess the status of the proposed algorithm, we compare it with several advanced learning-based methods. As shown in Table 7, the performance was excellent in terms of latency, throughput, and resource utilization. However, the algorithm has certain limitations, such as the need for high-performance GPU support during the training phase. These characteristics show that although our method is very effective in practical applications, it still needs to be further optimized in terms of hardware requirements to reduce costs.

As can be seen from Table 8, the proposed algorithm outperforms other modern learning scheduling methods on all three key performance metrics. This shows that the proposed algorithm not only has significant advantages over traditional algorithms, but also performs well when compared with other learning-based methods.

To further verify the scalability of the proposed algorithm in larger networks and its real-time adaptability, additional experiments are performed. Specifically, we tested the algorithm's performance under dynamically growing or shrinking network topologies.

Table 9 shows the performance of the proposed algorithm under dynamic network topology change. The proposed algorithm can maintain relatively stable performance no matter the network scale is expanded or reduced. The experiment results indicate the validity and scalability of our method.

Table 8: Comprehensive performance comparison with modern learning scheduling methods.

Algorithm	Average Latency (ms)	Throughput (Mbps)	Average Resource Utilization (%)
Proposed Algorithm	10.5	9800	92
CNN-based Scheduling	12.0	9500	89
A3C	13.5	9400	88
PPO	14.2	9300	87

Table 9: Performance under dynamic network topology changes.

Network Scale Change	Average Latency (ms)	Throughput (Mbps)	Average Resource Utilization (%)
Expansion (N=10K -> N=100K)	10.5 -> 11.5	9700 -> 9900	92 -> 93
Reduction (N=100K -> N=10K)	11.5 -> 10.5	9900 -> 9700	93 -> 92

Table 10: Statistical analysis of performance under link failure.

Link Failure Percentage	Average Latency (ms) ± Standard Deviation	Throughput (Mbps) ± Standard Deviation	Average Resource Utilization (%) ± Standard Deviation
5%	11.0 ± 0.5	9750 ± 100	91 ± 1
10%	12.3 ± 0.7	9650 ± 150	90 ± 2
15%	13.6 ± 0.8	9550 ± 200	89 ± 2

Although the proposed algorithm performs well in terms of latency and resource utilization, the computational complexity of GNNs and hierarchical reinforcement learning can be large, especially in real-time network management scenarios. Specifically, the initial training phase requires high-performance GPU support to handle large amounts of data and complex model parameters. However, once the model training is complete, real-time decisions can be made on a normal CPU. A large amount of memory is needed to store model parameters and intermediate results in the training process, but the memory requirement is low in the reasoning stage. After optimization, the algorithm can meet the real-time requirements in practical applications. In terms of robustness and failure scenarios, experimental results dealing with link failures and topology changes have demonstrated the good robustness of the proposed algorithm. To further enhance the statistical analysis of the experimental results, we report the mean deviation and confidence intervals around the results. For example, the standard deviations of the performance metrics (such as average delay time, throughput and resource utilization) of the proposed algorithm are small under the link failure rates of 5%, 10% and 15%, which indicates that the proposed algorithm has good stability and robustness. These statistical analyses further confirm the reliability and consistency of the proposed algorithm in the face of network failures and topology changes.

It can be seen from Table 10 that the proposed algorithm can maintain high performance even in the case of network failure, and the standard deviation is small, indicating that it has good stability and robustness.

As shown in Table 11, parameter sensitivity analysis was performed to understand the impact of hyperparameters on performance in the DRL framework. A higher learning rate may lead to instability, while a lower learning rate may prolong convergence time. The optimal learning rate is about 0.001. Discount factors affect the importance of long-term rewards. The optimal discount factor is 0.95, which balances short-term and long-term rewards.

Table 11: Hyperparameter sensitivity analysis.

Hyperparameter	Optimal value	Performance variation range
Learning rate	0.001	± 10%
Reward discount factor	0.95	± 5%

4.4 Discussion

Through the comparative analysis of the above experimental results, our traffic scheduling algorithm based on deep reinforcement learning and graph neural network shows significant advantages in multiple dimensions. Firstly, the algorithm can maintain a low average delay time under different network sizes and traffic demand intensities, which is attributed to the intelligent decision-making mechanism of the algorithm, which can quickly adapt to network changes and achieve efficient resource allocation. Secondly, the algorithm also performs well in resource utilization, especially in high traffic demand environment, not only maintains low latency, but also achieves near-optimal throughput, which shows that the algorithm not only improves network performance, but also pays attention to rational utilization of resources and avoids resource waste. However, any algorithm has its limitations. Although the algorithm demonstrated excellent performance in this study, its computational resource requirements are high, especially when dealing with large-scale networks. This is because the training process of deep reinforcement learning and graph neural networks is complex and requires a lot of computing and storage resources. Moreover, the real-time performance of the algorithm is also a problem worth considering, especially when the network scale expands sharply, the decision time of the algorithm may increase, which affects the efficiency of real-time scheduling.

Therefore, in the future research, optimization algorithm computational efficiency and real-time performance will be one of the important directions.

Our algorithm outperforms Dijkstra, Shortest Path First (SPF), and Weighted Round Robin (WRR) algorithms in terms of average latency, throughput, and resource utilization. Dijkstra algorithm and SPF algorithm are effective in finding the shortest path, but lack of dynamic perception of network state when dealing with complex network environment, resulting in low resource utilization. Although WRR algorithm is more uniform in resource allocation, it is not flexible and adaptive enough in the face of network congestion, and can not effectively reduce delay or improve throughput. In contrast, our algorithm, by combining deep reinforcement learning with graph neural networks, is able to better understand and predict the state of the network, thus making more optimized decisions and exhibiting more comprehensive advantages.

To sum up, our traffic scheduling algorithm based on deep reinforcement learning and graph neural network shows excellent performance in a variety of network scenarios, especially when dealing with large-scale networks and high traffic demand, the algorithm can maintain low latency, high throughput and efficient resource utilization. Despite the limitation of high computational resource requirements, further technological innovation and optimization are expected to overcome this challenge and enable the algorithm to be applied in a wider network environment. Future research will focus on real-time improvement of algorithms, optimization of computational efficiency and expansion of network security, in order to contribute to building a more intelligent, efficient and secure network system.

Although the proposed algorithm has significant performance advantages, the use of Graph Neural Networks (GNNs) and hierarchical reinforcement learning also brings an increase in computational complexity. The initial training phase requires significant computational resources and data support, which can be a trade-off. Specifically, GNNs require high computational resources when dealing with large-scale networks, especially in the training phase; hierarchical reinforcement learning similarly requires a large number of iterations to converge to the optimal policy. In addition, in order to train an effective model, the algorithms require a large amount of historical and simulated data, which may be difficult to obtain in practical applications. Although the algorithms can be trained to make quick decisions in real-time environments, the real-time responsiveness of the algorithms may be compromised in certain extreme situations, such as sudden large-scale traffic or frequent network failures.

The proposed algorithm shows superior performance in scenarios such as network failures, changes in traffic demand, and topology changes. In the case of network failure, DRL enables the algorithm to learn how to reroute traffic in case of link failure, reducing the delay due to a single point of failure. The GNN is able to identify critical nodes and paths in the network so that it can quickly find alternative paths in the event of a failure to ensure the continuity and stability of the network. For changes in traffic demand, the adaptive nature of DRL enables the algorithm to dynamically adjust resource allocation according to the current traffic demand. When the traffic demand increases, the algorithm can intelligently allocate more bandwidth to high-priority data streams while maintaining low latency. GNN helps the algorithm to understand the dependencies between different nodes for more effective traffic scheduling. In terms of topology changes, GNN handles changes in network topology well because it can predict the best paths by learning the connection patterns between nodes. Even when the network topology changes, the algorithm can quickly adapt to the new structure and find the optimal traffic scheduling scheme.

Together, these factors enable the proposed algorithm to remain efficient and stable in the face of complex and changing network environments. This novel approach not only improves the overall performance of the network, but also enhances its reliability and robustness in practical applications. Future research can further optimize the algorithm to reduce the computational overhead and improve its practicality.

5 Conclusion

In this paper, an intelligent solution based on deep reinforcement learning and graph neural network (GNN) is proposed for traffic scheduling problem in large-scale dynamic network environment. The algorithm innovatively combines GNNs and hierarchical reinforcement learning framework, and realizes efficient decision-making from macro strategy to micro-operation. Experimental results show that compared with traditional algorithms, the proposed algorithm achieves significant advantages in key performance indicators such as average delay time, throughput and resource utilization, especially in the face of complex scenarios such as network scale expansion, network congestion and network topology change, the robustness and generalization ability of the algorithm are fully verified. In addition, we optimize the learning efficiency of the algorithm through model initialization strategy, parameter adjustment method and convergence analysis, and ensure its stability and efficiency in dealing with large-scale network environment.

The results of this study not only provide new theoretical references and technical means for the field of network traffic scheduling, but also open up new research directions for automatic network traffic management. In the future, we plan to further explore the application of the algorithm in more complex network environments, including multimodal data fusion, cross-domain network scheduling and real-time network anomaly detection, with a view to contributing more to the development of future network technologies. Through continuous optimization and expansion of algorithm functions, we believe that the proposed intelligent traffic scheduling scheme will play a more important role in future network technology applications, laying a solid foundation for more efficient and intelligent network resource management.

References

[1] Cavone G, van den Boom T, Blenkers L, Dotoli M, Seatzu C, De Schutter B. An MPC-based rescheduling algorithm for disruptions and disturbances in large-scale railway networks. IEEE Transactions on Automation Science and Engineering. 2022; 19(1): 99-112. https://doi.org/10.1109/tase.2020.3040940

[2] Chen J, Chen J, Guo K. Queue-aware service orchestration and adaptive parallel traffic scheduling optimization in SDNFV-Enabled cloud computing. IEEE Transactions on Cloud Computing. 2023; 11(4): 3525-40. https://doi.org/10.1109/tcc.2023.3294239

[3] Chen JY, Wang Y, Ou JT, Fan CY, Lu XY, Liao CHS, et al. ALBRL: Automatic load-balancing architecture based on reinforcement learning in software-defined networking. Wireless Communications & Mobile Computing. 2022; 2022. https://doi.org/10.1155/2022/3866143

[4] Chung JM, Jo SW, Ahn MH. ARQ supportive EDF scheduling for wireless networks servicing real-time applications. IEEE Communications Letters. 2013; 17(7): 1329-31. https://doi.org/10.1109/lcomm.2013.052013.130261

[5] Coleman M, Tarte L, Chau S, Levine B, Reddy A. A data-driven approach to prioritizing bus schedule revisions at new york city transit. Transportation Research Record. 2018; 2672(8): 86-95. https://doi.org/10.1177/0361198118796717

[6] Corman F, Quaglietta E. Closing the loop in real-time railway control: Framework design and impacts on operations. Transportation Research Part C-Emerging Technologies. 2015; 54: 15-39. https://doi.org/10.1016/j.trc.2015.01.014

[7] Dabiran F, Rabiee HR, Salehi M, Amidian AA. A traffic-aware scheduling algorithm for IEEE 802.16 mesh mode. Scientia Iranica. 2014; 21(3): 803-14. https://doi.org/

[8] Dai B, Li HT, Wang YF. SP-DG: A programmable packet-level scheduling for queuing delay guarantees in time-critical networks. Computer Networks. 2024; 250. https://doi.org/10.1016/j.comnet.2024.110614

[9] Ganesan E, Hwang IS, Liem AT, Ab-Rahman MS. SDN-Enabled FiWi-IoT smart environment network traffic classification using supervised ML models. Photonics. 2021; 8(6): 201. https://doi.org/10.3390/photonics8060201

[10] Giridhar A, Kumar PR. Scheduling automated traffic on a network of roads. IEEE Transactions on Vehicular Technology. 2006; 55(5): 1467-74. https://doi.org/10.1109/tvt.2006.877472

[11] Hu WB, Wang H, Yan LP, Bo D. A hybrid cellular swarm. optimization method for traffic-light scheduling. Chinese Journal of Electronics. 2018; 27(3):611-6. https://doi.org/10.1049/cje.2018.02.002

[12] Jiang YB, Qiu ZL, Zhang MS, Li J. Integration of unicast and multicast scheduling in a two-stage switch architecture with low scheduling overhead. Iet Communications. 2012; 6(17): 2825-32. https://doi.org/10.1049/iet-com.2012.0105

[13] Lee SK, Shah SAR, Seok W, Moon J, Kim K, Shah SHR. An optimal network-aware scheduling technique for distributed deep learning in distributed HPC platforms. Electronics. 2023; 12(14):3021. https://doi.org/10.3390/electronics12143021

[14] Li T, Wang NM, Zhang M, He ZW. Dynamic reversible lane optimization in autonomous driving environments: balancing efficiency and safety. Journal of Industrial and Management Optimization. 2024; 20(3): 901-25. https://doi.org/10.3934/jimo.2023108

[15] Lian YD, Yang QF, Xie W, Zhang LW. Cyber-physical system-based heuristic planning and scheduling method for multiple automatic guided vehicles in logistics systems. IEEE Transactions on Industrial Informatics. 2021; 17(11): 7882-93. https://doi.org/10.1109/tii.2020.3034280

[16] Liao B, Pei J, Yang S, et al. Single-machine and parallel-machine parallel-batching scheduling considering deteriorating jobs, various group, and time-dependent setup time. Informatica, 2018; 29(2):281-301. https://doi.org/10.15388/Informatica.2018.168

[17] Janiak A, Krysiak T, Trela R. A Problem of scheduling jobs with non-monotonic stepwise values. Informatica, 2014; 25(1): 37-53. https://doi.org/10.15388/Informatica.2014.03

[18] Khemaissia I, Mosbahi O, Khalgui M, Li ZW, Qu T. Coherence and Feasibility of Real-Time Software Tasks in Networked Adaptive Systems. IEEE Access. 2018; 6: 35824-43. https://doi.org/10.1109/access.2018.2845942

[19] Kleinmann A, Wool A. Automatic construction of statechart-based anomaly detection models for multi-threaded industrial control systems. ACM Transactions on Intelligent Systems and Technology. 2017; 8(4): 1-21. https://doi.org/10.1145/3011018

[20] Liao DY, Wang CN. Neural-network-based delivery time estimates for prioritized 300-mm automatic material handling operations. IEEE Transactions on Semiconductor Manufacturing. 2004; 17(3): 324-32. https://doi.org/10.1109/tsm.2004.831533

[21] Lin WS, Sheu JW. Metro Traffic regulation by adaptive optimal control. IEEE Transactions on Intelligent Transportation Systems. 2011; 12(4): 1064-73. https://doi.org/10.1109/tits.2011.2142306

[22] Liu L, Shibasaki R, Zhang Y, Kosuge N, Zhang MY, Hu Y. Data-driven framework for extracting

global maritime shipping networks by machine learning. Ocean Engineering. 2023; 269: 113494. https://doi.org/10.1016/j.oceaneng.2022.113494

[23] Maniatis SI, Nikolouzou EG, Venieris IS. End-to-end QoS specification issues in the converged MAP wired and wireless environment. IEEE Communications Magazine. 2004;42(6):80-6. https://doi.org/10.1109/mcom.2004.1304236

[24] Ouyang LQ, Lam WHK, Li ZC, Huang D. Network user equilibrium model for scheduling daily activity travel patterns in congested networks. Transportation Research Record. 2011; (2254):131-9. https://doi.org/10.3141/2254-14

[25] Radzi NAM, Suhaimy N, Ahmad W, Ismail A, Abdullah F, Jamaludin MZ, et al. Context aware traffic scheduling algorithm for power distribution in smart grid network. IEEE Access. 2019; 7: 104072-84. https://doi.org/10.1109/access.2019.2931722

[26] Raubenheimer H, Engelbrecht A. A Division-of-Labour approach to traffic light scheduling. Applied Sciences-Basel. 2024; 14(17): 8022. https://doi.org/10.3390/app14178022

[27] Sciortino JC. Autonomous ESM systems. Naval Engineers Journal. 1997; 109(6): 73-84. https://doi.org/10.1111/j.1559-3584.1997.tb01947.x

[28] Shan T, Yang O, Zhang GZ. A traffic scheduling framework in broadband wireless access systems. Computer Communications. 2003;26(14):1602-13. https://doi.org/10.1016/s0140-3664(03)00060-4

[29] Shor J, Robertazzi TG. Traffic sensitive algorithms and performance-measures for the generation of self-organizing radio network schedules. IEEE Transactions on Communications. 1993;41(1):16-21. https://doi.org/10.1109/26.212359

[30] van der Heijden MC, van Harten A, Ebben MJR, Saanen YA, Valentin EC, Verbraeck A. Using simulation to design an automated underground system for transporting freight around Schiphol Airport. Interfaces. 2002; 32(4): 1-18. https://doi.org/10.1287/inte.32.4.1.49

[31] Wang X, Li I, Wang D, Zhuang HQ, Morgera SD. Incorporating retransmission in quality-of-service guaranteed multiuser scheduling over wireless links. IEEE Transactions on Vehicular Technology. 2009;58(8):4388-97. https://doi.org/10.1109/tvt.2009.2021983

[32] Wang Y, Qiu DW, Strbac G. Multi-agent deep reinforcement learning for resilience-driven routing and scheduling of mobile energy storage systems. Applied Energy. 2022; 310: 118575. https://doi.org/10.1016/j.apenergy.2022.118575

Abbreviations

Abbreviation	Full name	Description
GNNs	Graph Neural Networks	A type of neural network designed to handle graph-structured data, capable of modeling complex relationships between nodes.
DRL	Deep Reinforcement Learning	A machine learning approach that combines deep learning and reinforcement learning techniques, enabling agents to learn optimal policies through trial and error in complex environments.
A3C	Asynchronous Advantage Actor-Critic	A reinforcement learning algorithm based on asynchronous parallel execution, aimed at improving training efficiency and stability.
PPO	Proximal Policy Optimization	An improved policy gradient method designed to address the high variance issues in traditional policy gradient methods while maintaining good convergence.
N	Number of Nodes in the Network	A parameter representing the scale of the network, typically used to describe the size or complexity of the network.