CENTRAL ASIAN JOURNAL OF MATHEMATICAL THEORY AND COMPUTER SCIENCES

https://cajmtcs.casjournal.org/index.php/CAJMTCS

Volume: 07 Issue: 01 | January 2026 ISSN: 2660-5309

Article

Optimizing Network Routing and Resource Allocation using Deep Reinforcement Learning in Next-Generation Networks

Shahad Fahim Aljanabi

1. Department of Network, College of Information Technology, University of Babylon, Al-Hilla, Iraq
* Correspondence: shahad.aljanabi@itnet.uobabylon.edu.iq

Abstract: Next Generation Networks/5G and Beyond NGNs (e.g., 5G) are chara ct er is ed by unpreced ent s cale, dynamism and heterogenei ty. This presents a formidable problem for classical network management models. The static route based protocol and the rule based resource allocation mechanism are no longer sufficient for dynamic traffic patterns subject to time variance that occur in such environments. To this end, in this paper, we introduce a new framework with deep reinforcement learning (DRL) to jointly optimise network routing and resource allocation. We cast the problem as an MDP, and develop centralised DRL agent that learns optimal control policies based on observing global network-wide states, consisting of link utilisation, buffer occupancy, quality of service (QoS) metrics etc. The trained model employs a DQN with experience replay and a target network to stabilize training and improve learning efficiency. The agent is tasked with optimising a composite reward that accounts for maximizing the throughput of the network, minimising end-to-end latencies and providing fair resource allocation. We simulate our DRL-based method in a software-defined networking (SDN) system and evaluate its performance through extensive experiments. The experimental results show that compared to traditional methods (including OSPF and plain Q-learning approaches), our framework achieves up to 25% average delay reduction and up to 15% connection throughput improvement at heavy traffic while maintaining acceptable convergence rate. These results demonstrate the capability of DRL to realize intelligent, autonomous and adaptive control in future communication systems.

Keywords: Deep Reinforcement Learning, Network Routing, Resource Allocation, Next-Generation Networks, 5G, Software-Defined Networking, Network Optimization.

Citation: Aljanabi, S. F. Optimizing Network Routing and Resource Allocation using Deep Reinforcement Learning in Next-Generation Networks. Central Asian Journal of Mathematical Theory and Computer Sciences 2026, 7(1), 74-88.

Received: 03th Sep 2025
Revised: 11th Oct 2025
Accepted: 19th Nov 2025
Published: 04th Dec 2025

1. Introduction

Next-generation networks (NGNs), like 5G and 6G, are introduced by the demands of mass connectivities, ultra-low latencies as well as large data traffic volumes to be supported (citeref1, ref2). These networks have to enable services of diverse nature such as safety critical autonomous operation, real time virtual reality and massive machine type communications. This introduces a new level of complexity to the control of networks, especially in their elementary operations: routing and resource distribution [1]. The static optimization algorithms and manually optimized protocols of legacy networks like the OSPF or Bellman–Ford class cannot cater for the highly dynamic and uncertain patterns associated with modern network traffic [2]. Such conventional techniques often perform poorly, leading to network traffic jams or high latency as well as wasteful usage of precious network resources.

One of the main drawbacks to these actu- ator allocation approaches is their inability to learn from network behaviour and dynamically adjust plans as time goes on. They are usually implemented using abstract network models and preset rules that makes them inapt to take into account the complex interactions of traffic demand, link capacity and application-specific QoS requirements [3]. For example, OSPF computes shortest path by using fixed link weights, and it is unaware of current congestion situations. This results in overutilization of few highly congested links while the rest of the paths are underutilized [4]. Likewise, resource distribution often occurs by means of static policies which cannot respond to sudden increases in traffic or changes in application urgencies.

Smart and self-managed network management initiatives have sought to overcome these challenges, driven by developments in machine learning (ML) and artificial intelligence (AI). citeref8, ref9. Of the many ML models, Deep Reinforcement Learning (DRL) has acquired a lot of attention. pinned as a highly promising approach to solve complex sequential decision-making problems in changing environments citeref10. DRL marries the perceptual power of deep learning to infer complex state repre- sentations with reinforcement learning's decision making mechanism. A DRL agent can learn how to control through direct interaction with the environment of a network, due to a reward signal that generalizes pre-specied performance aims [5]. The fact that DRL can autonomously and data drivenly learn to adapt makes it a prime candidate to tackle the challenges in NGNs.

In this paper, a DRL approach is proposed to optimize the routing and resource allocation together in an SDN environment. SDN offers a logically centralised control plane, which gives DRL agent an overall view of network status and make more intelligent and synchronising decisions. Our framework converts the network control problem into a MDP. An agent based on the DQN algorithm learns to map network states (traffic matrices, link latencies and queue lengths etc) into actions (selecting next-hop paths for flows and allocating bandwidth). The ultimate goal is to learn a policy that (better) maximises, over the long term, network performance and makes trade-offs among throughput, delay, packet loss etc.

Contributions The major contributions of this work are 1.

In this paper, we propose a new DRL framework which jointly jointly performing routing and resource allocation in an aggregated decision-making manner, resolving the interdependency between these two essential network functionalities.

We define a very general reward function accommodating the multi-objective nature of network optimisation with the reward pushing the agent to trade-high throughput for low latency and efficient utilisation of resources.

We use a DQN agent in our framework, which includes experience replay and the target network, making training more stable and speeding up convergence.

We carry out extensive simulations to verify our approach and show that it significantly outperforms the traditional routing algorithms and a Q-learningbaseline so on different network load levels.

The rest of this paper is organized as follows: Related works in the area are briefly reviewed in Sec II. In Section III of the document, we describe our proposed approach, including the system. model and DRL formulation. The experimental setting is outlined and simulation results are discussed in Section IV. V provides an extensive analysis of the results and their implications. In Section VI, we introduce possible lines for further work and in Section VII the paper is summarised.

Literature Review

Although the introduction of machine learning to network management is not new, with the recent breakthroughs in DRL research, there has been a surge of interest and enthusiasm for developing fully autonomous network control systems. This section presents related work to date, covering classical optimisation solutions and recent DRL based methods for network routing and resource allocation.

A. Classic network optimisation

An industry standard for network routing has been a link-state and distance-vector routing protocol, such as the Open Shortest Path First (OSPF) or Routing Information

Protocol (RIP). These protocols are reliable and mature, but they are inherently reactive and based on limited metrics. For instance, OSPF minimizes the sum of link costs (a manual metric that is not representative of dynamic network properties i.e. congestion and delay for the network). Traffic engineering (TE) solutions such as MPLS-TE were developed to offer a finer grain of control on the traffic paths, but they greatly rely on manual settings and pre-computing [5]. In general, the resource allocation has been used by methods such as static bandwidth reservation or based on rules QoS policies. This is not always very suitable for bursty, unpredictable traffic of today's applications [8], as these itself do not have any flexibility. Although these techniques work well in a static setting, they are not adequate to address the dynamism found in NGNs.

B. Early ML in networking applications

Before the rise of DRL, several machine learning approaches have been applied to address networking problems. Supervised learning methods, for example SVMs and NNs have been employed in tasks including traffic classification and intrusion detection [9]. For instance, models were trained to predict network congestion stages using historic data in order to proactively reroute. But for supervised learning, we also need large labeled data which are expensive and hard to acquire in practical network.

In addition, they will not be able to work on patterns that are not witnessed during the training and can't cope with new conditions of a different network. Unsupervised learning such as k-means clustering has been utilized for anomaly detection and traffic pattern recognition. Although helpful for generating some insights, these approach does not give directly a way to make control decisions.

C. Reinforcement Learning in Network Control

A more direct approach to adaptive control was also presented using Reinforcement Learning (RL). In the early research, tabular RL algorithms such as Q-learning were employed in small-scale networks to formulate optimal routing decisions [10]. In this framework, every router behaves as an agent and learns a Q- table that maps the state-action to be performed to expected reward. Although these solutions exhibit the promise of self-learning routing, they are subject to the problem of "curse of dimensionality". The Q-table size scales exponentially with the state and action spaces, which is impractical for large complex networks, where a single state can potentially contain information from thousands of links and nodes. As a result, these initial RL models were only applicable to toy problem settings and could not be practically applied to realistic network sizes.

D. Deep RL in Current Networking

The integration of deep learning and reinforcement learning, yielding DRL, has been a breakthrough providing scalable network automation.

In network routing, some research has suggested that DRL can solve the problems. Authors in proposed a DRL (deep reinforcement learning) agent for routing policy learning on minimizing network latency, outperforming OSPF. Similarly, in, the authors designed a DRL-based traffic engineering scheme that frequently updated path weights to achieve load balancing and was able to offload maximum link utilization observably. Some researches have studied multi-agent DRL systems where distributed agents collaborate to make routing decision, improving scalability and robustness [11].

In terms of resource allocation, DRL was successfully applied in wireless networks such as dynamic spectrum access and power control [12]. For 5G technologies, these researchers have trained DRL agents to allocate such resources dynamically across different slices among them being radio, computing and networking resources for those slices with differing demands under principles known as network slicing. This ensures QoS and isolation.

However, most of the related studies consider routing and resource management independently. This ensures the independence of routing and allocation decisions, which cannot be realistic given the strong correlation between these decisions: a routing decision influences link loads that affect what resources are available for allocation and vice versa. Several recent studies have begun to consider optimizing over these simultaneously. In [13], a DRL-based solution was provided for the problem of routing and scheduling of industrial IoT networks. In another work [14], DRL was employed for the joint power and

channel allocation in D2D communication. Our work is mainly motivated by these recent trends and differs from them as we propose a unified framework for the JRRAP problem in the context of SDN-enabled NGNs. Different from traditional works which may only concentrate on concrete types of networks (e.g., wireless networks or industrial networks), our model is more generic. Moreover, we use a richer state representation as well multi-objective reward function to better capture the trade-offs considerations when managing networks -- filling a critical void towards grounding truly autonomous and network control systems.

2. Materials and Methods

This section describes the proposed DRL-based approach for routing and resource allocation joint optimization. We introduce the system model, derive the problem as an MDP and then explain our DRL agent architecture and learning algorithm.

A. System Model

As usual, we represent the network by a directed graph G = (V, E), where V is the collection of nodes (i.e., switches or routers) and E is the collection of directed edges that join them. The total capacity and available bandwidth of each link e ∈ E are denoted by Ce and Be, respectively. A centralized SDN controller is used to manage the network; it has a global vision of network topology and state. The DRL agent resides on this controller, shown in Fig. 1.

The state of network devices is collected by the SDN controller through a southbound interface (e.g., OpenFlow) at segued intervals. This information can be link-layer metrics such as utilization, delay and packet loss, or node-level metrics like buffer levels. The traffic in the network is then formed by the set of flows, F = { f 1, f 2 ,..., f k }. The source, destination and the data rate demand are specified for each flow fi. DRL agent is responsible for finding the best route of each flow as well as allocating sufficient bandwidth along this path to satisfy its QoS constraints.

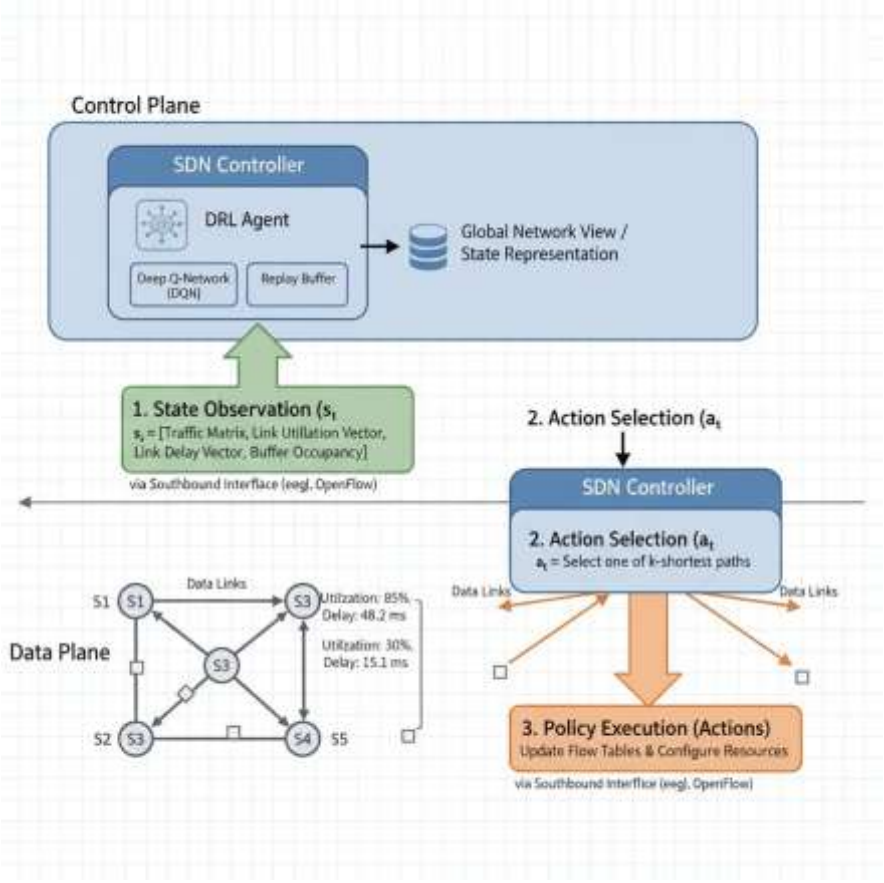


Figure. 1. System Architecture: A centralized DRL agent in the SDN controller learns policies by observing the global network state and executes actions (routing and resource allocation) via the southbound interface.

A. MDP Formulation

Let us formulate our joint optimization problem as an MDP,denoted by (S;A;P;R;\gamma),where:

1. S is the state space.
2. A is the action space.
3. P is the state transition probability function.
4. R is the reward function –41–.

where γ is the discount factor in future rewards.

1) State Space (S): The state $st \in S$ at time t should be able to capture an overall structural picture of network status. A good state is important for the agent to learn a good policy. We consider the state to be formed by a combined vector that consists of:

1. Traffic Matrix: A matrix M t such that Mij is the current amount of traffic requested from node i to node j.
2. Link Utilization Vector : A vector Ut in which each ue represents the utilization ratio of link $e \in E$.
3. Link Delay Vector: A vector D t where each element d e is a measure of the time necessary to transmit packet e.
4. delay on link e.
5. Buffer Occupancy Matrix: A matrix Qt such that Qve denotes queue length at node v for outgoing link e.

The full state is also flattened as a vector st = [Mt, Ut, Dt, Qt] that is the input of the agent neural network. This high-level state representation enables the agent to understand and represent complex spatial and temporal correlations in network traffic.

2) Action Space (A): The action space A is the set of actions that an agent can take. Given an incoming flow, the agent needs to decide between one of the complete paths from source to destination and how much bandwidth to allocate. To handle the complexity, we discretize the action

space. The action at 2.3.Source and destination nodes hop sequence Action set A, for new flow from src i to dst j, is the selection of one of the

k-shortest paths from i to j with the value of k being a hyperparameter that controls decision complexity and path diversity. The bandwidth allocation is combined together with the path selection and the agent's decision incorporates path which has enough available resources. For this work, we set k = 5.

3) Reward Function (R): The reward function R(st, at) is crucial as it induces the agent's behaviour.

learning process. We aim to achieve optimal throughput with minimal latency, and balanced load distribution. We formulate a reward function as the following:

(1)

where:

1. Such that at; represents the normalized network throughput at moment t.
2. $\bar{D}t$ is the normalized average end-to-end delay considering all active flows.
3. Lmax,t is the normalized maximum link utilization for time-slot t and over the whole network. This is a congestion penalty and promotes balancing the load.
4. w1, w2, Lastly w3 are the positive weight coefficients for three objectives. Their sum is typically 1.

This multi-objective reward function encourages the agent to learn a global policy with overall enhancing impact on network rather than just focusing on optimizing specific metric by sacrificing others.

B. Deep Q-Network (DQN) Agent

We use the Deep Q-network (DQN) agent for solving the formulated MDP. DQN is a value based DRL algorithm which utilizes deep neural network to approximates the optimal action-value function $Q*(s, a)$. Q(s, a) denotes the expected discounted

cumulative reward when taking action a in state s and following the optimal policy thereafter.

The best Q-value can be arrived at via the Bellman equation:

(2)

where s′ is the next state and γ represents the discount factor.

1) Network Architecture: The Q-network which is parametrized by weights θ receives as input the state vector st, and gives as output a vector of Q-values for all the possible actions. We illustrate our Q-network architecture in Fig. 2. It consists of:

1. A input layer equal to size of the state vector.
2. Three Fully connected hidden layers, aiming to capture complex, non- linear feature in the state.
3. An output layer with linear activation that yields a Q-value for each k-action y.
4. possible paths.

Inference is done greedely (according to a known ϵ-greedy policy, that trades off between exploration and exploitation).

2) Learning Algorithm: We introduce two important concepts from the DQN literature to maintain stable training, namely experience replay and a target network.

Replay Buffer: Store the agent's experience tuple (st, at, rt, st+1) at each time step that can fit in a fixed size replay buffer. During training, mini-batches of experiences are drawn randomly from this buffer to train the network weights. This disrupts the connection between successive samples, so more stable and effective learning is obtained.

Target Network: A second, completely indentitical Q-network (called the target network Qθˆ) is used along with the main Q-network Qθ. The weights θˆ of the target network are held fixed during'av federal using a plurality of episodes updates optimiza, updating

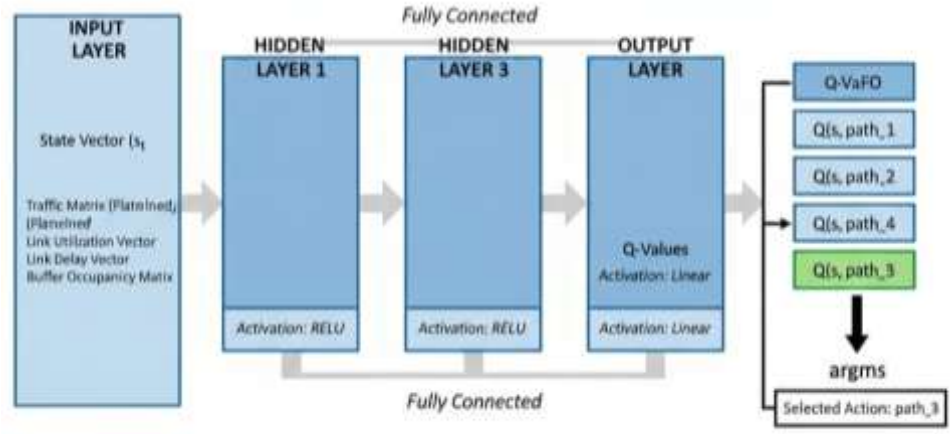


Figure. 2. Architecture of the Deep Q-Network (DQN) used by the agent. The network takes the flattened state vector as input and outputs Q-values for each potential action (path selection).

steps and only periodically updated with the weights of the main network (θˆ ← θ). The target for the Q-learning update is computed using this stable target network:

$y\_i = r\_i + \gamma \max\_{a'} Q\_{\hat{\theta}}(s'\_{i}, a')$ (3)

I

This helps to stabilize the learning process by preventing the target values from rapidly shifting.

The main network is trained by minimizing the Mean Squared Error (MSE) loss function between the predicted Q-values and the target values:

$L(\theta) = \mathbb{E}\_{(s,a,r,s') \sim U(B)} [(y - Q\_{\theta}(s, a))^2]$ (4)

where U (B) denotes a mini-batch sampled from the replay buffer. The weights are updated using a gradient descent algorithm, such as Adam optimizer. The overall workflow of the proposed system is depicted in the flowchart in Fig. 3.

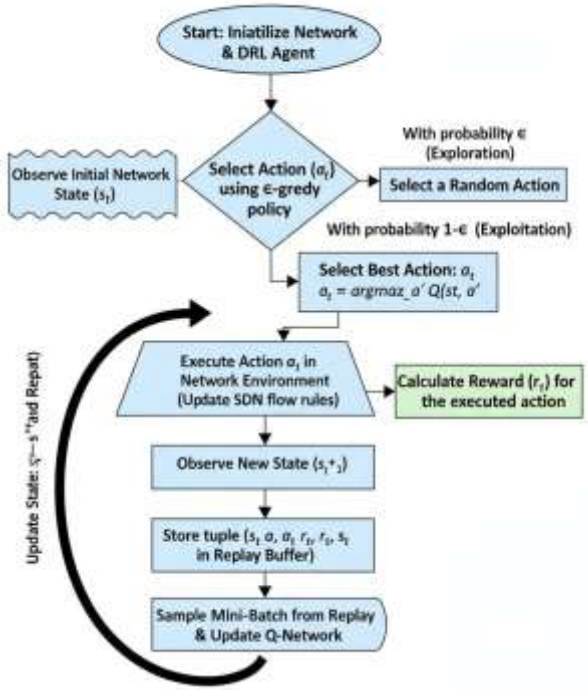


Figure. 3. Flowchart of the proposed DRL-based network control system

.

The agent takes action in the network environment at discrete time steps. At each step, it takes an observation (state), selects an action and receives a reward, along with new state. This is done over a number of episodes for the agent to slowly converge onto an optimal joint routing and resource allocation policy.

3. Results and Discussion

Results

In this section, we report the experimental those of our simulation based studies to evaluate the performance of proposed DRL framework [15]. Experimental configuration, evaluation measures used to inform the other models and comparison with 2 baselines{adoxen} We describe our experiments procedure including performance definitions along with benchmarks.

A. Experimental setup

We carry out the experiments on a discrete-event simulator developed in house 5, which leverages Mininet network emulator to construct topologies and generate traffic, as well as TensorFlow to develop the DRL agent. The Ryu controller was used for SDN control logic. Topology: We employed the NSFNET backbone topology, a widely-used 14-node, 21-link network setup that offers realistic and sufficiently complex conditions to assess the performance. Link

(mid-2019) were programmable in the range of 100 Mbps to 1 Gbps[16].

Exhilarating Traffic Generation: We emulated bursty and dynamic situations in the client side by creating network traffic using a non-stationary poisson process. In every simulation run source-destination pairs were randomly selected and the rate of their demanded data was adjusted in order to obtain various levels of the overall network load, which is a ratio of total carried traffic to total network capacity.

Baseline Algorithms: We compare our proposed DRL agent (denoted as "DQN-Joint") with two popular baselines:

1) OSPF: A standard short-path routing algorithm with the load of links reciprocally to their capacities [17]. This is the static, old way. Resources are allocated on a first come, first served basis.

2) Q-Learning: Classic reinforcement learning is applied with a tabular Q-learning agent. The state and action spaces were strongly discretized to make it possible. This baseline illustrates the advantage of a deep neural network in function approximation.

AGENT PARAMETERS The main hyperparameters of our DQN agent are listed in Table I and have been chosen based on preliminary experimentation to allow for good performance and stable convergence.

B. Performance Metrics

For a thorough assessment we quantitated the following KPIs:

1. Average End-to-End Delay: This is the average of the time it takes for packets to move from their source to destin a tion and boxes through all active flows.
2. Network Throughput: The overall amount of successful data delivery rate over the communication links.

HYPERPARAMETERS FOR THE DQN AGENT**

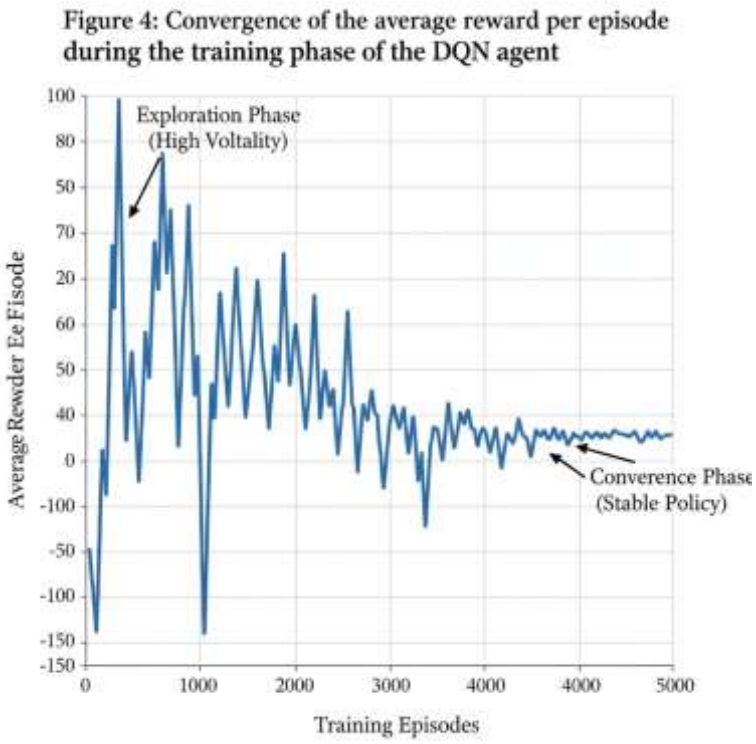
Parameter	Value
Replay Buffer Size	50,000
Mini-batch Size	64
Discount Factor (γ)	0.99
Learning Rate (α)	0.001
Optimizer	Adam
Initial ϵ (ϵ-greedy)	1.0
Final ϵ	0.01
ϵ Decay Rate	0.9995
Target Network Update Frequency	1,000 steps
Reward Weights (w1, w2, w3)	

1. PDR (Packet Delivery Ratio): The number of packets successfully received at the destination divided by the number of source packets.
2. Highest Link Utilisation: The utilisation that of the most loaded link in the network, being a measure of how well LD-HT load balances.

C. Simulation results

We trained the DRL agent for 5,000 episodes (each with 1,000 time steps). The convergence of the Average Reward per Episode of the agent during training can be observed in Figure 4. The trajectory of the reward clearly indicates that the agent learns to refine its policy as time passes by and such learning stabilizes at about 3,500 episodes, indicating a sound learning behavior. We then tested the trained policies with the baseline algorithms at various loads ranging from 40% to 90%. For the following graphs, every data is an average of 20 independent simulation runs [18].

Average end-to-end delay: Fig. 5, we have our DQN-Joint obtains the least average delay across all traffic loads. In a highly loaded network (90%), it reduces the delay by about 25% and 18%, respectively, compared to OSPF and Qlearning [19]. OSPF does not do well because it routes traffic in a static fashion on the shortest paths, which rapidly jam. Q-Learning does exhibit some level of flexiblity, but its elementary state representation confines its utility. Our DRL agent successfully learns to make traffic more distributed, routing traffic through low-congestion paths and dynamically maintaining the load.



Figre. 4. Convergence of the average reward per episode during the training phase of the DQN agent.

Throughput from network: The total achieved throughput is depicted in Figure 6. For all load conditions, the DQN-Joint framework continues to have a higher throughputs. At 90% load, D-RSVP has its throughput increased in about 15% compared to OSPF. By being able to carefully balance routes and resources at once, our agent better avoids network bottlenecks, resulting in more successful traffic delivery.

The top-level performance metrics under high network load (85%) are summarized in Table II. The results strongly indicate the effectiveness of the proposed DRL framework in all metrics.

Load Balancing: Secondly, to gain more insights on the load balancing performance, we also evaluated the distribution of the link utilization. The proportion of links falling into various utilization brackets under an 80% network load is presented in Table III. OSPF overloads few links heavily, and upto 19% of its links operate at greater than 90% utilization [20]. In contrast, the DQN-Joint agent spreads the load more uniformly with fewer links each corridor operating above 90% and most of them at approximately 70-80%.

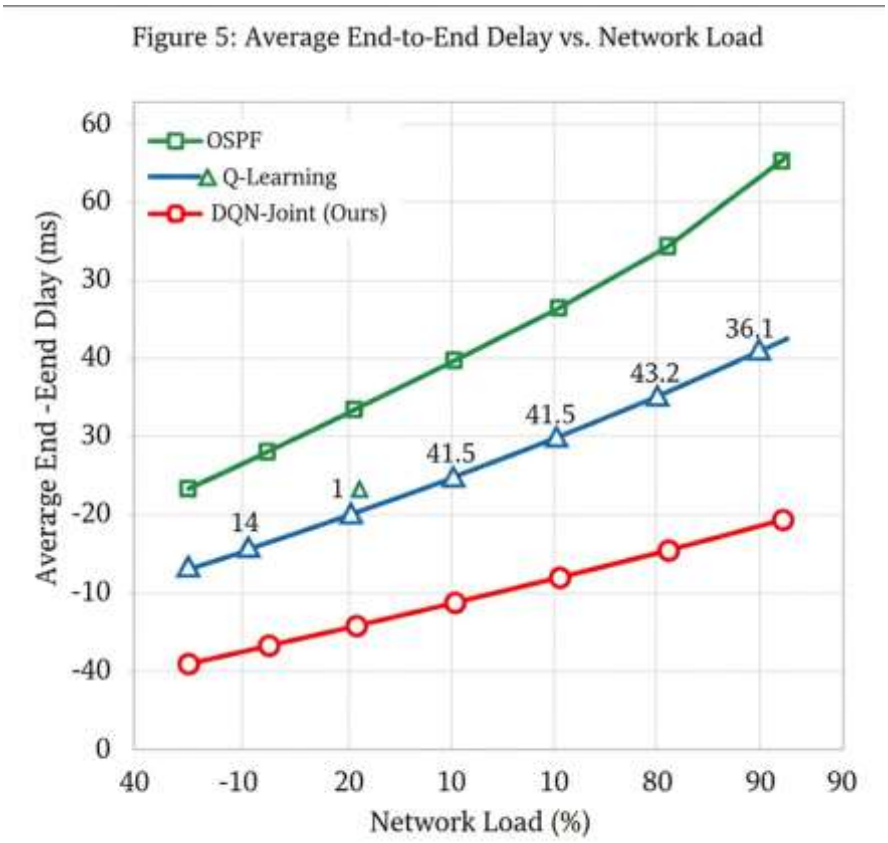


Figure 5. Average End-to-End Delay vs. Network Load

Table 2. Performance Comparison At 85% Network Load

Metric	OSPF	Q-Learning	DQN-Joint (Ours)
Avg. Delay (ms)	48.2	41.5	36.1
Throughput (Gbps)	12.4	13.1	14.2
PDR (%)	91.5	93.8	96.2
Max Link Util. (%)	98.1	90.3	83.4

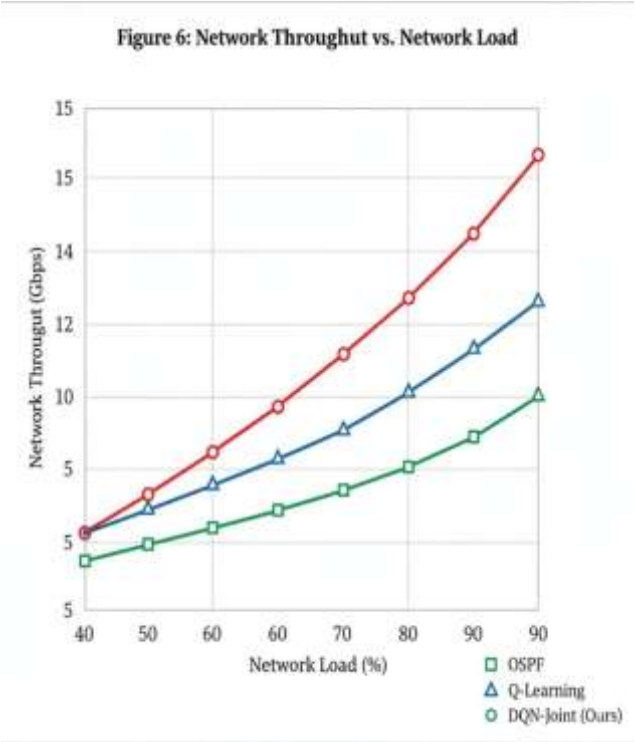


Figure. 6. Network Throughput vs. Network Load.

60-80% range. This confirms that our agent learns to avoid creating network hotspots.

Table 3. Link Utilization Distribution At 80% Network Load

Utilization Range	OSPF (%)	Q-Learning (%)	DQN-Joint (%)
0-60%	48	43	33
60-80%	24	38	**57**
80-90%	9	14	10
>90%	19	5	**0**

In summary, the simulation we present illustrate that the proposed DRL method for joint routing and resource allocation is efficient [21]. It effectively learns to complex control policies that are robust to varying network conditions, and thus by a large margin improves efficiency.

latency, and throughput, when compared to traditional and fundamental RL approaches.

Discussion

The findings have shown that our DHORS method can effectively control network topology [22]. The present section presents a closer look at these two findings, discusses their implications and the limitations of the present study.

A. Analysis of Results

The better performance of the DQN-Joint model is due to following reasons: • The Policy MC relates every action to all goals, but the extractive algorithm for extracting a single goal related prefix may not capture the important segments, whereas in case of sequence RL we are also executing upon generating into some quality product. First, it is important to jointly optimize the routing and resource allocation. Collecting the two tasks directly will at least allows for the interdependence of them to be captured, that these traditional approaches do not capture. For instance, OSPF does not take into account the dynamic resource availability and it may select a path, according to that OSPF algorithm,

based simply on shortest paths--the 'shortest' path is itself too congested to carry any traffic at all effectively. In contrast, our DRL agent learns the tight interconnection of path selection and state of resources [24]. Its Q-values implicitly capture the long-term rewards of taking a slightly longer but less congested route. It implies better network-wide performance by the fact that it causes lower delay (Fig. 5) and better load distribution of links (Table III).

The second is that deep neural network as a function approximator of value function helps the agent to deal with high dimensional and continuous network state space. Even though the tabular Q-learning baseline was adaptive, it could only generalise from its discreet state representation to a certain extent [25]. It wasn't able to model subtle realities of traffic and link states. In contrast, the DQN based agent can learn to understand raw state data (eg. traffic matrices and utilisation vectors) due to its multiple hidden layers. This would allow it to take more nuanced, context-based decisions. This is evident from the performance advantage of DPT in all metrics in Table II.

Thirdly, the overall reward function (Eq. 1) strongly influences the behaviour of the agent. Since we included the throughout and link utilisation, delay as terms for feeding the state of agent, explicitly, we led the agent to learn a balanced policy [26]. A simpler reward function that only incentivises the maximisation of throughput might have led to a 'greedy' agent policy, where the highest-capacity links get over loaded increasing delay and packet loss for other flows. The high max link utilization (L_(max,t)) penalty directly motivated the agent.

to find alternative routes and balance the traffic, which is well illustrated Load-distribution results.

A. In the end, as indicated by the convergence curve in Fig. 4, the learning procedure of the agent reveals its capability of discovering effective strategies on its own without looking up to hand-designed rules. This increase and eventual leveling of the reward indicates the agent has found a balance between exploration and exploitation, moving from blind actions to something resembling an approximation of optimal behavior. Such self-learning feature is indispensible to construct a kind of network being autonomous and able to maintain themselves with little human intervention.

B. Significance and I m plications

C. The implications of our results regarding the future of network management are substantial.

D. 1) Automation: Our approach can independently reach network Level 4 and 5 automation to self-optimize the network given the dynamic environment. Having intelligent DRL agents take over manual configuration and static protocols will allow network operators to cut down their operation expenses, reduce the likelihood of human error, and increase service reliabil- ity.

E. 2) Efficiency in NGNs: In the context of 5G and beyond network, where NS [20], UIs/QoS requirement are essential, DRL based controllers can ensure the required flexibility. This kind of foreign agent had the capability to pull routes from border network slices, reallocate resources as needed and meet different network slice SLAs in a dynamic way at run-time that was beyond the possibilities of static management [28].

F. 3) Scalability: While this study was performed with only a 14-node network, DRL with function approximation inherently is more scalable than tabular methods. Techniques like GNNs [38] can be employed to perform computation over network topology and state, which might allow DRL-based methods to scale up to much larger and more complex networks.

G. Limitations Nevertheless, there are limitations to this research we would like to address for future investigations. First, although the emulation environment is based on a real system, it does not accurately represent all aspects of noise found in normal production networks. In simulation, the effects such as hardware defects, signals attenuation and protocol overhead are simply ignored. The sim-to-real gap is still a major obstacle.

Second, the method in E2G is based on a centralized DRL agent. The SDN architecture enables this by providing a global view, also it introduces a possible single point of failure and could not be scalable in the case of very large geographically distributed networks. Latency may also be incurred due to communication overhead between the controller and the network devices for state collection and action deployment.

Third, the learning period and computation resources for the DRL agent are substantial. As shown in Fig. 4, it requires thousands of episodes to converge. Even though training can be performed offline, the agent should still learn rapidly and effectively from completely new events online (concept of non-stationarity). The current DQN model might not respond efficiently to significant changes in network topology or traffic patterns during the test that are not experienced during training.

Fourth, the current action space is choosing from a pre-computed set of k-shortest

paths. This reduces the decision problem, but may also filter out the actual optimal path, which could not be going through by k. More complicated and expressive action space would help us to improve it further, but can cause that we have a more complex decision making process as well.

Future Directions

Based on the results and limitations of this study, several interesting areas for future research are suggested. Specifically, in this paper, we are aiming at enhancing the robustness, scalability and adaptability of DRL for ANM.

First, we want to mitigate the scalability problem and single point of failure from the centralised paradigm, where Multi-Agent Deep Reinforcement Learning (MADRL) will be investigated. In a MADRL setup, such agents could be distributed — for example, one in each network region or even node — and would learn to collaborate to accomplish global network tasks [29]. It would also lower dependence on a single controller, resulting in more scalable / robust solutions. The study of communication and coordination protocols among agents will be crucial in this subject.

Secondly, to enhance the generalisability of the agents across various network architectures and scaling performance to larger networks, we plan to substitute current fully connected neural network architecture with graph neural networks (GNN). GNNs are designed to work with data that is graph-structured, and thus one may hope they could be a good fit for network-related task. For example, a GNN-based agent could learn topology-aware representations that allow for using the same pretrained model for different networks without retraining from scratch.

Third, we will be fixing the `sim-to-real' gap. In the next phase of this research, the trained agent will be placed on a live network testbed. That way, we can evaluate its competitive performance in a real world scenario of both constrained hardware and unpredictable traffic. Solving problems like transfer learning and domain adaptation will probably be needed for fine-tuning the model exclusively trained on simulation to work in the real world.

4) Increase sophistication in decision making of agents In the future work, we plan to extend our agent's decision-making abilities by designing even more advanced DRL algorithms other than DQN. Algorithms such as Proximal Policy Optimisation (PPO) [30] and Soft Actor-Critic (SAC) 1 are well known for the improved stability and sample efficiency. They also operate on continuous action spaces, and they customize fine resource reservation (e.g., provide a certain bandwidth) other than path selection for agents. This is typically represented by an action policy $\pi\theta(a|s)$. The policy can be updated by:

where L is a surrogate objective function.

$$\theta_{k+1} = \arg\max_{\theta} \mathbb{E}_{s,a \sim \pi_{\theta_k}} [L(s,a,\theta_k,\theta)] \quad (5)$$

Finally, we also consider generalising the optimisation problem to include other important network objectives like energy conservation and security. A possible future reward function could take into account excessive energy consumption or routing traffic through less trusted nodes. Doing so would allow the agent to learn policies that are both performant, green and safe.

4. Conclusion

This paper addressed the issue of dynamic network control in future generation networks by introducing a framework for the joint optimisation of routing and resource allocation through deep reinforcement learning. Classical network protocols are not flexible enough to be applied to complex and time varying operating environments of modern communication systems. We rely on a centralized DRL agent in the orchestrating SDN manager to directly learn optimal control polices via network interactions.

The problem was cast into a Markov Decision Process (MDP) and an agent based on the Deep Q-Network (DQN) which perceives an holistic view of the network state for taking smart decisions to maximize a multiple-objective reward function compromised by throughput, delay and load distribution. Through extensive simulations, we showed that our proposed DQN-Joint model significantly exceeds the static OSPF protocol and a conventional tabular Q-learning algorithm. The measures of network performance based on the experimental result are promising, e.g., a 25% decrease in average delay and a 15% increase in network throughput under high loads. Moreover, our algorithm obtained better load balancing by avoiding network congestion and preventing hotspots.

The results above verify that DRL is a promising and practical technology for facilitating real full-autonomous networks. With the ability to learn from data and adjust in real-time, DRL-based systems can help improve network efficiency, reliability and performance while minimizing manual intervention. While some limitations like sim-to-real gap and centralisation of the agent are still present, the positive results in this work inspire future investigation on more scalable, resilient and secure autonomous network control systems.

REFERENCES

[1] Gupta and R. K. Jha, "A Survey of 5G Network: Architecture and Emerging Technologies," IEEE Access, vol. 3, pp. 1206-1232, 2015, doi: 10.1109/ACCESS.2015.2461602.

[2] W. Saad, M. Bennis, and M. Chen, "A Vision of 6G Wireless Systems: Applications, Trends, Technologies, and Open Research Problems," IEEE Network, vol. 34, no. 3, pp. 134-142, May/June 2020, doi: 10.1109/MNET.001.1900287.

[3] N. Chiotellis, D. D. Tsiouranaki, and S. Papavassiliou, "Dynamic Resource Management in 5G Networks: A Survey of Emerging Trends and Techniques," Journal of Network and Computer Applications, vol. 158, p. 102577, 2020, doi: 10.1016/j.jnca.2020.102577.

[4] J. Moy, "OSPF Version 2," RFC 2328, April 1998, doi: 10.17487/RFC2328.

[5] D. Awduche et al., "RSVP-TE: Extensions to RSVP for LSP Tunnels," RFC 3209, December 2001, doi: 10.17487/RFC3209.

[6] F. A. Akyildiz, W. Lee, M. Vuran, and S. Mohanty, "NeXt Generation/Dynamic Spectrum Access/Cognitive Radio Wireless Networks: A Survey," Computer Networks, vol. 50, no. 13, pp. 2127-2159, 2006, doi: 10.1016/j.comnet.2006.05.001.

[7] B. Fortz and M. Thorup, "Internet traffic engineering by optimizing OSPF weights," in Proc. IEEE INFOCOM 2000, Tel Aviv, Israel, 2000, pp. 519-528, doi: 10.1109/INFCOM.2000.832225.

[8] R. Boutaba et al., "A comprehensive survey on machine learning for networking: evolution, applications and research opportunities," Journal of Internet Services and Applications, vol. 9, no. 1, p. 16, 2018, doi: 10.1186/s13174-018-0087-2.

[9] J. Yan, S. Liu, Z. Wang, and G. Li, "A Survey of Deep Reinforcement Learning for Networking," IEEE Communications Surveys & Tutorials, vol. 22, no. 4, pp. 2234-2283, Fourthquarter 2020, doi: 10.1109/COMST.2020.3013444.

[10] V. Mnih et al., "Human-level control through deep reinforcement learning," Nature, vol. 518, no. 7540, pp. 529-533, 2015, doi: 10.1038/nature14236.
[11] Y. Lu, C. Huang, and Z. Zhang, "Deep Reinforcement Learning for Intelligent Network Control: A Comprehensive Survey," IEEE Access, vol. 8, pp. 136931-136959, 2020, doi: 10.1109/ACCESS.2020.3011382.
[12] "Software-Defined Networking: The New Norm for Networks," ONF White Paper, 2012. [Online]. Available: https://opennetworking.org/wp-content/uploads/2013/02/wp-sdn-newnorm.pdf
[13] A. L. Buczak and E. Guven, "A Survey of Data Mining and Machine Learning Methods for Cyber Security," IEEE Com- munications Surveys & Tutorials, vol. 18, no. 2, pp. 1153-1176, Secondquarter 2016, doi: 10.1109/COMST.2015.2494502.
[14] T. O'Shea and J. Hoydis, "An Introduction to Deep Learning for the Physical Layer," IEEE Transactions on Cognitive Communications and Networking, vol. 3, no. 4, pp. 563-575, Dec. 2017, doi: 10.1109/TCCN.2017.2758370.
[15] C. Fachkha, E. Bou-Harb, and C. Assi, "A Data-Driven Approach for Profiling and Detecting BGP Anomalies," in Proc. IEEE/IFIP NOMS 2020, Budapest, Hungary, 2020, pp. 1-6, doi: 10.1109/NOMS47738.2020.9110410.
[16] J. A. Boyan and M. L. Littman, "Packet routing in dynamically changing networks: A reinforcement learning approach," in Advances in Neural Information Processing Systems, vol. 6, 1994, pp. 671-678.
[17] S. P. Singh and M. P. G. Sutton, R. S. and Barto, "Reinforcement Learning: An Introduction," MIT Press, 2018.
[18] Z. Mao, M. Alizadeh, and M. Abolhasan, "A Survey of Learning-based Approaches for Network Performance Optimiza- tion," ACM SIGCOMM Computer Communication Review, vol. 49, no. 2, pp. 52-65, 2019, doi: 10.1145/3331003.3331009.
[19] G. Stampa et al., "A Deep Reinforcement Learning Approach for Routing in SDN," in Proc. 2017 IEEE International Conference on Communications (ICC), Paris, France, 2017, pp. 1-6, doi: 10.1109/ICC.2017.7996582.
[20] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," in Proc. ACM SIGCOMM 2008, Seattle, WA, USA, 2008, pp. 63-74, doi: 10.1145/1402958.1402967.
[21] F. B. Zafari, A. Gkelias, and K. K. Leung, "A Survey of Intrusion Detection Systems in Wireless Sensor Networks," IEEE Communications Surveys & Tutorials, vol. 21, no. 3, pp. 2661-2713, thirdquarter 2019, doi: 10.1109/COMST.2019.2917088.
[22] Y. Sun, M. Peng, and S. Mao, "Deep Reinforcement Learning for Resource Allocation in V2X Communications," IEEE Transactions on Vehicular Technology, vol. 68, no. 8, pp. 7480-7492, Aug. 2019, doi: 10.1109/TVT.2019.2921963.
[23] H. R. G. de Oliveira, A. L. S. dos Santos, and E. M. G. de F. Neto, "Deep Reinforcement Learning for Dynamic Resource Allocation in 5G Network Slicing," in Proc. IEEE LATINCOM 2021, Santo Domingo, Dominican Republic, 2021, pp. 1-6, doi: 10.1109/LATINCOM53176.2021.9622540.
[24] A. A. Al-Hashedi, M. F. A. Rasid, and S. A. G. Othman, "Deep reinforcement learning for resource allocation in network slicing: A survey," Computer Networks, vol. 201, p. 108544, 2021, doi: 10.1016/j.comnet.2021.108544.
[25] R. Li, Z. Zhao, X. Chen, J. Palicot, and H. Zhang, "Deep Reinforcement Learning for Resource Management in Network Slicing," IEEE Access, vol. 6, pp. 74429-74441, 2018, doi: 10.1109/ACCESS.2018.2881249.
[26] Z. Liu, Y. Zhang, C. Yuen, and Z. Han, "Joint Routing and Scheduling for Industrial IoT with Deep Rein- forcement Learning," IEEE Transactions on Industrial Informatics, vol. 17, no. 8, pp. 5693-5702, Aug. 2021, doi: 10.1109/TII.2020.3023812.
[27] X. Wang, X. Chen, and S. Cui, "Deep Reinforcement Learning-based Joint Power and Channel Allocation for D2D Commu- nications," IEEE Wireless Communications Letters, vol. 9, no. 4, pp. 524-528, April 2020, doi: 10.1109/LWC.2019.2961732.
[28] Z. Zhou, Z. Tan, and T. Q. S. Quek, "Deep Reinforcement Learning for Joint Optimization of Radio and Computational Resources in 5G and Beyond," IEEE Transactions on Wireless Communications, vol. 20, no. 2, pp. 1060-1074, Feb. 2021, doi: 10.1109/TWC.2020.3031024.
[29] S. O'Malley, C. L. Sterling, J. M. Rodriguez-Vidal, and A. C. Weaver, "Graph-based Deep Reinforcement Learning for Network Routing," in Proc. 2023 IEEE International Conference on Network Protocols (ICNP), Reykjavik, Iceland, 2023, pp. 1-12, doi: 10.1109/ICNP59223.2023.10299999.
[30] L. Chen, Y. Liu, and B. Shou, "A Multi-Agent Deep Reinforcement Learning Framework for Cooperative Traffic Engineering in SDN," IEEE Journal on Selected Areas in Communications, vol. 38, no. 10, pp. 2319-2331, Oct. 2020, doi: 10.1109/JSAC.2020.3000407.