Packet Routing Against Network Congestion: A Deep Multi-agent Reinforcement Learning Approach

Ruijin Ding*, Yuwen Yang*, Jun Liu†, Hongyan Li‡, and Feifei Gao*
*Department of Automation, Tsinghua University, Beijing, China
†Institute for Network Sciences and Cyberspace, Tsinghua University, Beijing 100084, China
Beijing National Research Center for Information Science and Technology (Tsinghua University), Beijing 100084, China
‡State Key Laboratory of Integrated Service Networks, Xidian University, Xian, 710071, China
Email:{drj17@mails.tsinghua.edu.cn, yyw18@mails.tsinghua.edu.cn, juneliu@tsinghua.edu.cn, hyli@xidian.edu.cn, feifeigao@ieee.org}

***Abstract*—The continuous growth of the network data would lead to the increased network congestion and the throughput decline. In this paper, we investigate the packet routing problem based on deep multi-agent reinforcement learning, where each router chooses the next hop router by itself intelligently. We design the modified deep Q-network in each router to evaluate the neighbor routers. The routers, each acting as an agent, choose the next hop router based on their local observation. Then they transfer the packets to the chosen routers and receive the reward and the observation of the next hop routers. Using their experience, the routers learn to improve the packet routing strategy by updating their Q-networks. We demonstrate that with proper reward set and training mechanism, the routers in the network can work in a distributed way to reduce the computational complexity compared with the single-agent reinforcement learning based algorithm. And the proposed algorithm can further reduce the congestion probability and improve the network performance.**

I. Introduction

The development of communication technologies brings the tremendous growth of network traffic [1]. In the era of 5G [2] and beyond, the high data rate [3], massive device connectivity [4], and the growth of data amount burden the heterogeneous backbone networks, which leads to the increased risk of network congestion and poses a severe challenge to the existing routing strategies.

Specifically, traditional rule-based strategies choose paths according to one metric value. They do not make full use of the abundant traffic patterns and may not be suited to the more and more complicated network. For example, the shortest path algorithms (e.g., OSPF [5], RIP [6], and so forth), which aim for the low latency, choose the transmission path based on the path length and will cause severe network congestion under heavy network traffic. Conversely, the network congestion would increase the delay and reduce the network performance.

In order to exploit the traffic patterns and balance the various demands in the network, machine learning based intelligent network traffic control strategies have drawn much attention in academia. Due to the rapid development of Soft Defined Routers (SDRs), the complex neural network operations can be integrated with programmable routers using CPUs and GPUs [7]. The researchers can apply deep learning [8] methods to the routing strategy and design intelligent routing protocol.

The authors of [9] demonstrate how the GPU-accelerated SDRs enable the deep learning technique to compute the routing path for the network packet transmission. This work is in the supervised learning manner, i.e., it runs the traditional routing protocols to get the training data. However, the supervised learning based method is essentially an imitation of traditional routing strategy. Following [9], the deep convolutional neural network is utilized to judge whether the corresponding path combination could cause network congestion [10]. However, with the increase of network size, training a neural network for each path combination leads to exponential growth in the number of required networks, and this method needs a centralized controller to choose the path combinations for the packets.

Meanwhile, reinforcement learning (RL) [11] is inherently suitable for learning strategies. A major advantage of RL over supervised learning is that RL does not rely on instructional information, and thus it does not need to label data manually. The Q-routing proposed in [12] modifies Q-learning [13] to apply it for routing. However, Q-learning cannot solve the problems with large state space, and thus Q-routing is not applicable for massive traffic patterns in the network. Furthermore, the authors of [1] use deep reinforcement learning (DRL) to improve the network performance in terms of network congestion probability and transmission path length. However, [1] still needs a centralized controller to instruct the routers to transfer the packets since it is based on single-agent reinforcement learning.

In this paper, we consider the packet routing problem in network with heavy traffic. The objective is to design a packet routing strategy that minimizes the congestion probability and meanwhile shortens the transmission path length. In order to

This work was supported by Tsinghua University Initiative Scientific Research Program 2019Z08QCX19 and Beijing Municipal Natural Science Foundation under Grants 4182030 and L182042.

978-1-7281-4905-9/20/$31.00 ©2020 IEEE

eliminate the dependence on the centralized controller, we utilize the deep multi-agent reinforcement learning (MARL) technique to design a distributed algorithm where each router chooses the next hop router according to the local observation. The simulation results demonstrate that through an appropriate reward set and training, the routers can learn from the interaction with the network environment and build an intelligent routing strategy to cooperate with each other in a distributed manner.

II. Preliminary

RL [11] is a learning method where an agent learns from interaction with the environment. The agent observes the environment state and chooses action based on it. Then the environment reacts to the action and feedbacks the reward signal to the agent. In RL, the objective of the agent is to find a policy to maximize the expected cumulative reward. The policy $\pi(a|s)$ represents the probability of choosing action a when the state is s. This goal is equivalent to finding the optimal state-action value function

$$q_*(s,a) = \max_{\pi} \mathbb{E}_{\pi}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \middle| S_t = s, A_t = a\right], \quad (1)$$

where $\mathbb{E}_{\pi}[\cdot]$ denotes expectation under policy π, R_t denotes the reward received at time step t, and γ is the discount rate. The optimal state-action value function $q_*(s,a)$ evaluates the value of action a under state s. Then the optimal policy is the greedy policy where the agent chooses the action with the highest value.

A. Q-Learning and Deep Q-Network

Q-learning [13] obtains the optimal state-action value function by iteratively updating, i.e.,

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\Big[R_{t+1} + \gamma \max_{a}\ Q(S_{t+1}, a) - Q(S_t, A_t)\Big] \quad (2)$$

where α is the step-size parameter.

However, Q-learning is a tabular method and can not handle the practical problems with large state space. In deep Q-network (DQN) [14], a deep neural network (DNN) is used to represent the state-action value function. DQN utilizes two tricks in training, i.e., experience replay and separate target network to break the correlation.

B. Multi-Agent Reinforcement Learning

In MARL [15] problem, multiple agents, denoted by L_i, $i = 1, \cdots, N$, interact with the environment together. As shown in Fig. 1, at each time step t, each agent L_i observes the environment, gets the observation $O_t^{(i)}$ that is determined by the state S_t, and chooses the action $A_t^{(i)}$ based on the observation. Then each agent receives the reward $R_{t+1}^{(i)}$ and the environment turns into S_{t+1}.

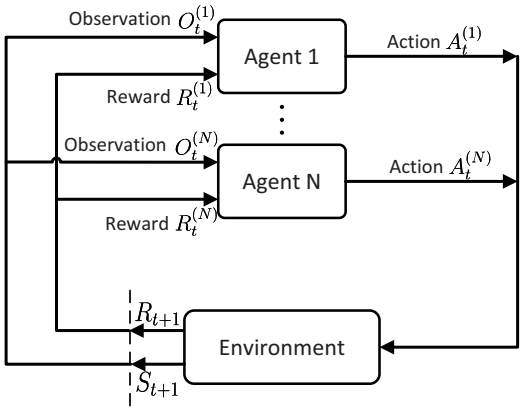

Fig. 1. Multi-agent reinforcement learning

III. System Model

Consider a general backbone network with N routers, as shown in Fig. 2. The set of routers is denoted by $\mathcal{L} = \{L_1, L_2, \cdots, L_N\}$. The routers include source routers, destination routers, and regular routers. Specifically, the source routers receive the packets from data sources, the destination routers are destinations of the packets, and the regular routers are assumed to forward the packets.

Moreover, we assume that the network operates in discrete time steps [10]. At each time step, each router selects the next hop router for the packets stored in its buffer and then transfers the packets. Each router has a limited buffer to temporarily store the received packets for later transmission. When the size of the received packet exceeds the remaining buffer size of the router, the network congestion occurs.

The traditional routing protocols can be formulated as searching a shortest path in a graph, where the data packets are transmitted along the shortest path. However, the shortest path based protocols may easily cause the network congestion. As shown in Fig. 2, the source routers L_0, L_1, and L_3 receive input packets and transmit them to the destination router L_8. According to the shortest path principle, all source routers choose L_4 to forward the packets to L_8. When the burst traffic appears at the source router, the remaining buffer size of L_4 will not be sufficient, which leads to the network congestion. In addition, when the similar burst traffic appears again, the traditional routing protocol would make the same decision and fall into the same congestion. Hence, the traditional routing protocols are not intelligent and cannot learn from the past experience to avoid the similar mistakes when facing similar scenarios.

IV. Deep MARL Based Packet Routing Algorithm

In the packet routing problem, all the routers cooperate with each other to transmit the data packets to their destinations. The routers can be naturally regarded as the agents, interacting with the network, and learn the routing strategy from the interaction experiences. For each agent, there is a neural

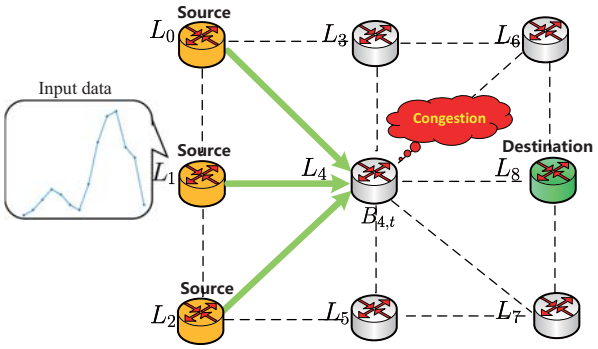

Fig. 2. The network topology

network to evaluate the connected routers. The agent chooses the one with the highest value as the next hop router.

The proposed deep MARL based packet routing algorithm can be divided into two phases, i.e., the centralized training and distributed implementation phases. In the centralized training phase, the neural network parameters for each router are accessible to other routers. Each agent would recur to other agents to update its own neural network parameters. In the distributed implementation phase, each agent observes the local environment and utilizes the well-trained neural network parameters to select the next hop router based on the observation. The detailed deep MARL based packet routing algorithm is as follows.

A. State and Observation Space

As mentioned in Section III, the congestion is determined by the remaining buffer size of the routers and the size of the packets. In addition, when making decision, the destination of the data packets ought to be considered. Therefore, the environment state should include the remaining buffer size of the routers as well as the size and destinations of the packets. However, each agent can only acquire the information from itself and the neighbor routers. The observation of agent i at time step t is

$$O_t^{(i)} = \{\boldsymbol{b}_t^{(i)}, p, d\}, \tag{3}$$

where the vector $\boldsymbol{b}_t^{(i)}$ is the remaining buffer size of L_i's neighbor routers. Moreover, the scalars p and d denote the size and destination of the packet respectively.

Besides, when the transferred packet causes the congestion or arrives at the destination, the transmission task of this packet terminates.

B. Action Space

In the packet routing problem, each agent selects the next hop router for the packets among its neighbor routers. As a result, the action space of each agent is naturally its neighbor routers. Each agent has different action space, as well as different neural network architectures, since the number of neighbor routers varies from router to router.

C. Reward Function

RL can transform difficult optimization problems into maximizing the expected cumulative reward problems through appropriate reward function design. In the packet routing problem, the objectives are divided into two parts: minimizing the congestion probability and shortening the transmission path.

To achieve the first goal, when network congestion happens, the environment should feedback the agent a negative reward r_c, which informs the agent that the action under the current observation is supposed to be avoided. To shorten the transmission path, there should be a fixed negative reward every time the packet is transferred, and we set it to -1. When the packet arrives its destination router, environment would give a non-negative reward, which is set to 0. If the discount rate γ is set to 1, then greater cumulative rewards means lower network congestion probability and shorter transmission path. Therefore, the reward function can be set as

$$R_{t+1} = \begin{cases} r_c & \text{if network congestion occurs,} \\ 0 & \text{if packet arrives destination,} \\ -1 & \text{otherwise,} \end{cases} \tag{4}$$

D. Specific Algorithm

We resort to DQN to implement the deep MARL based packet routing algorithm. Each agent L_i has a Q-network $Q_i(o^{(i)}, a; \theta_i)$ that takes its current observation $O_t^{(i)}$ as input and then outputs the value of each action, i.e., each neighbor router of L_i. At each time step, all agents observe the environment and then compute the action-value function based on the observations. The agent selects the highest value action, with a probability of $1-\epsilon$, i.e., ϵ-greedy. Following the environment transition caused by the actions, each agent i stores the transition $\left(O_t^{(i)}, A_t^{(i)}, R_{t+1}^{(i)}, O_{t+1}^{(j)}\right)$ in the corresponding replay memory $\mathcal{D}_i$, where $O_{t+1}^{(j)}$ is the observation of the selected next hop router L_j.

The update rule of the proposed algorithm is a little different from the standard DQN. In standard DQN update procedure, a random minibatch is sampled from the replay memory for updating in order to break the strong correlations between the samples. Then the stochastic gradient-decent (SGD) method is performed to minimize the loss function:

$$\left[R_{t+1} + \gamma \max_{a'} Q(S_{t+1}, a'; \theta^-) - Q(S_t, A_t; \theta)\right]^2, \tag{5}$$

where $R_{t+1}+\gamma \max_{a'} Q(S_{t+1}, a'; \theta^-)$ is the update target, and θ^- are the parameters of the separate target network. They are duplicated from the training parameters θ every fixed number of steps for further reducing the correlation between the target and the training part.

Let us modify the calculation of the update target in (5). To be specific, suppose that agent L_i chooses L_j as the next hop router for the packet, and then L_j feedbacks its observation as well as the ACK signal. The loss function is converted to

$$\left[R_{t+1}^{(i)} + \gamma \max_{a'} Q_j(O_{t+1}^{(j)}, a'; \theta_j) - Q_i(O_t^{(i)}, A_t^{(i)}; \theta_i)\right]^2. \tag{6}$$

In fact, Q_i evaluates the path length from L_i to the destination (opposite number). When the packet is transferred from L_i to L_j, $R_{t+1}^{(i)} + \gamma \max_{a'} Q_j(O_{t+1}^{(j)}, a'; \theta_j)$ is used as the better estimate of the path length. Moreover, using the estimate from other agents to compute the update target can break the correlation between the target and the training part. As a result, there is no need to set a separate target network. The overall algorithm is summarized in Algorithm 1.

Algorithm 1 Packet Routing with Deep MARL

Initialize the whole system, including the buffer size of all the routers, the replay memory $\mathcal{D}_1, \mathcal{D}_2, \cdots, \mathcal{D}_N$, and the action-value functions with random parameters $\theta_1, \theta_2, \cdots, \theta_N$.
for each time step t **do**
  for each agent L_i **do**
    for each packet in L_i **do**
      Observe the information of L_i's neighbor routers, and attach the size and destination of the packet
      Choose action $A_t^{(i)}$ with ϵ-greedy
      The next hop router L_j feedbacks its observation $O_{t+1}^{(j)}$ as well as the ACK signal
      Store $\left(O_t^{(i)}, A_t^{(i)}, R_{t+1}^{(i)}, O_{t+1}^{(j)}\right)$ in the replay memory $\mathcal{D}_i$
    end for
    Sample random minibatch of transitions from $\mathcal{D}_i$
    Perform a gradient descent step on (6) with respect to the network parameters θ_i
  end for
end for

V. Simulation Results

In this section, simulations are conducted to evaluate the performance of the deep MARL based packet routing algorithm. We compare the performance of the proposed algorithm with Q-routing [12], DOMT-DQN [1], and traditional shortest path based routing protocol, e.g., OSPF. For better illustration, we consider the simple network with topology depicted in Fig. 2.

A. Complexity Analysis

Since the proposed deep MARL algorithm is in a distributed manner, the computation of each agent can be greatly reduced compared with the centralized one. Based on the description in Section IV, the input and output layers of the neural network in each agent are determined by its neighbors. The distributed way simplifies the computation for each agent, and hence, only one hidden layer would achieve good enough performance. The specific network architectures are shown in Table I.

TABLE I
The Neural Architectures

Agent	Input layer	Hidden layer	Output layer
L_4	10	12	8
$L_0 \sim L_8$ except L_4	5	7	3

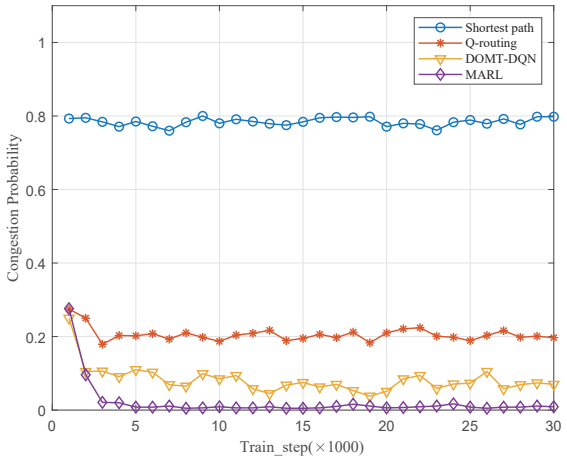

Fig. 3. Network congestion probability comparison with training steps.

The required number of floating point operations (FLOPs) is used as the metric of computational complexity [16]. Then the total FLOPs of all neural networks is summarized in Table II.

TABLE II
Complexity Comparison

Algorithm	Total FLOPs
MARL in this paper	1.2×10^3
DOMT-DQN in [1]	6.1×10^3

Compared to the centralized algorithm DOMT-DQN, the distributed algorithm can significantly reduce the computational complexity. Obviously, Q-routing [12] has the least computational complexity since it is a tabular method. However, when facing the diverse data packets, the performance of Q-routing is unsatisfactory, and will be seen later.

B. Performance Comparison

The buffer size of each router is set to 45 MB, and the packets are discarded when the buffer is full. The time step is set to 10 ms. If the network congests in a time step, we mark it and then compute the congestion probability as the proportion of time steps that are congested in every 1000 time steps.

In training phase, we assume that at each time step, the size of newly generated packets obeys the Poisson distribution, whose mean values are also random variables and obey discrete uniform distribution of $[9, 29)$ MB. In this way, the algorithm can deal with more diverse scenarios in the implementation phase.

In Fig. 3, we compare the congestion probability of the proposed deep MARL algorithm, DOMT-DQN, Q-routing, and the shortest path protocol. From the figure, all three learning based methods can significantly reduce the congestion probability compared to the shortest path baseline and converge quickly. Among them, the congestion probability of the

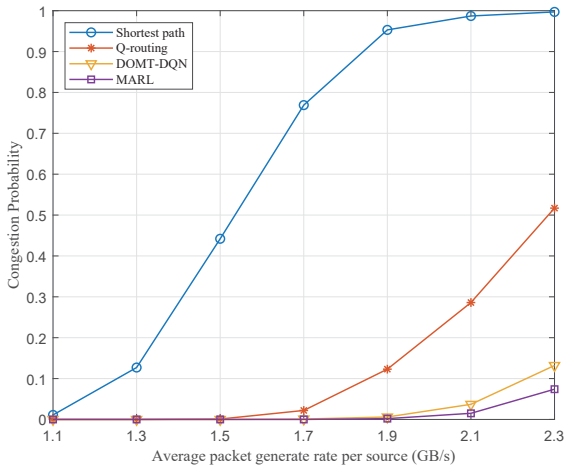

Fig. 4. Network congestion probability comparison for various packet generation rates.

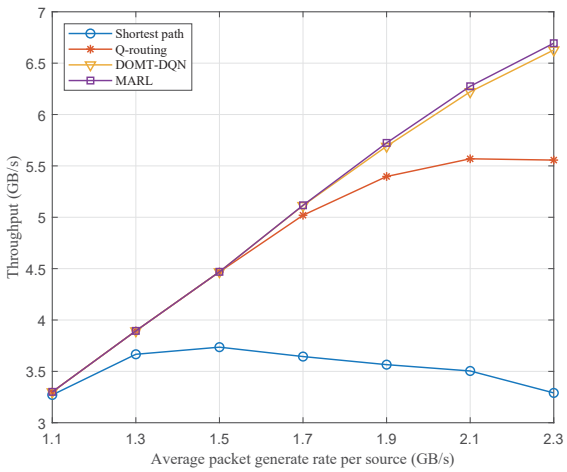

Fig. 5. Network throughput comparison for various packet generation rates.

proposed deep MARL algorithm decreases the most, followed by DOMT-DQN, while Q-routing decreases the least. The reason can be explained as followed: (i) Q-routing only utilizes the index information of the routers without considering the actual network state. When facing the network with diverse input packets, the index based Q-routing cannot respond in time; (ii) Although DOMT-DQN considers the network state information, it suffers from the invalid actions problem, which may influence its network congestion avoidance and path shortening. The centralized approach has significantly larger action space than the distributed approach and thus enhances the difficulty of action selection.

To better compare the performance differences between these algorithms, we select well-trained neural network parameters (well-trained Q-tables for Q-routing) to run these learning based algorithms 1000 time steps, i.e., the implementation phase. We plot in Fig. 4 the congestion probability versus different packet generation rates. When the packet generation rate is low, all methods perform well in the network. But the congestion probability of the baseline method increases significantly with the growth of the packet generation rate. In contrast, the proposed deep MARL algorithm suppresses the increase of congestion probability as the packet generation rate grows. Similarly, the proposed deep MARL algorithm always outperforms the other two methods, demonstrating its robustness against packet generation rate variation. Remarkably, when the packet generation rate is lower than 2 GB/s, the proposed deep MARL algorithm maintains 0% congestion probability. Even with more than 2 GB/s packet generation rate, the congestion probability is lower than 0.1.

Fig. 5 shows the throughput of the network under different packet generation rates. Due to increasing congestion probability, the shortest path baseline may suffer from a slightly decline in throughput even though more data enters into the network. The proposed deep MARL algorithm achieves the best performance as evidenced in Fig. 4.

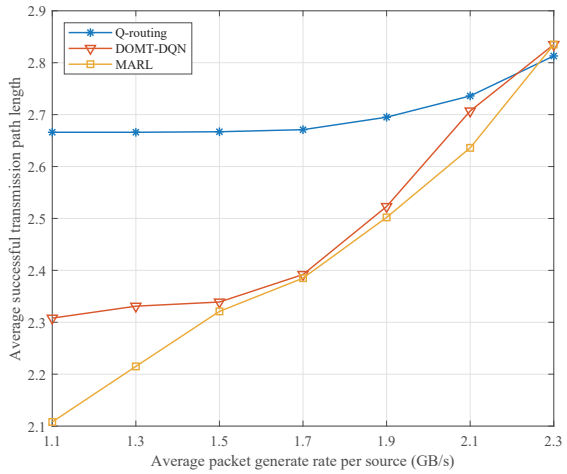

Fig. 6. Average successful transmission path length with varying packet generation rates.

We show in Fig. 6 the average successful transmission path length with varying packet generation rates. With low data rate, the proposed deep MARL algorithm almost achieves the shortest path, i.e., two hops, which is much shorter than the centralized algorithm DOMT-DQN. The increased packet generated rate requires longer data transmission path to maintain the low congestion probability. Interestingly, it seems that the path length of Q-routing can be even shorter than the proposed deep MARL algorithm with the rate achieving 2.3 GB/s. The reason is that the transmission path of Q-routing is fixed. Q-routing is an index based algorithm. When stop training, the well-trained Q-tables indicate fixed choices for each agent, leading to the fixed paths. The congested packets are discarded and will not be considered in the average successful transmission path length. As a result, the shorter

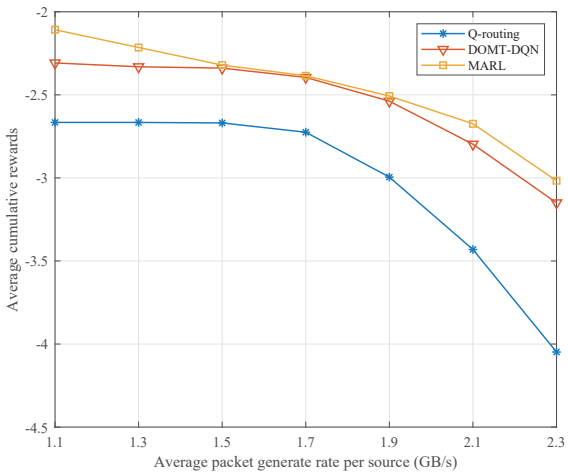

Fig. 7. Average cumulative reward with varying packet generation rates.

successful transmission paths are traded for higher congestion probability (about 0.5 with rate 2.3 GB/s).

The average cumulative rewards shown in Fig. 7 take into account both the transmission path length and the network congestion. From the figure, the average cumulative rewards drop for all algorithms since the network traffic becomes heavier with the growth of packet generation rate. The proposed algorithm outperforms DOMT-DQN, although the former consumes much fewer computational resources.

VI. Conclusion

In this paper, we developed a distributed packet routing algorithm based on deep MARL for network with heavy traffic. We demonstrate that in the training phase, the proposed algorithm converges fast to low congestion probability. In the implementation phase, the proposed algorithm outperforms other methods in terms of congestion probability, throughput, and path length.

References

[1] R. Ding, Y. Xu, F. Gao, X. Shen, and W. Wu, "Deep reinforcement learning for router selection in network with heavy traffic," *IEEE Access*, vol. 7, pp. 37 109–37 120, 2019.

[2] J. G. Andrews *et al.*, "What will 5g be?" *IEEE J. Sel. Areas Commun.*, vol. 32, no. 6, pp. 1065–1082, Jun. 2014.

[3] W. Roh *et al.*, "Millimeter-wave beamforming as an enabling technology for 5g cellular communications: theoretical feasibility and prototype results," *IEEE Commun. Mag.*, vol. 52, no. 2, pp. 106–113, Feb. 2014.

[4] T. S. Rappaport *et al.*, "Millimeter wave mobile communications for 5g cellular: It will work!" *IEEE Access*, vol. 1, pp. 335–349, May 2013.

[5] B. Fortz and M. Thorup, "Optimizing ospf/is-is weights in a changing world," *IEEE J. Sel. Areas Commun.*, vol. 20, no. 4, pp. 756–767, 2002.

[6] C. L. Hedrick, "Routing information protocol," United States, Jun. 1988. [Online]. Available: https://www.rfc-editor.org/info/rfc1058

[7] S. Han, K. Jang, K. Park, and S. Moon, "Packetshader: A gpu-accelerated software router," *SIGCOMM Comput. Commun. Rev.*, vol. 40, no. 4, pp. 195–206, Aug. 2010.

[8] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015.

[9] B. Mao *et al.*, "Routing or computing? the paradigm shift towards intelligent computer network packet transmission based on deep learning," *IEEE Trans. Comput.*, vol. 66, no. 11, pp. 1946–1960, Nov. 2017.

[10] F. Tang *et al.*, "On removing routing protocol from future wireless networks: A real-time deep learning approach for intelligent traffic control," *IEEE Wireless Commun.*, vol. 25, no. 1, pp. 154–160, Feb. 2018.

[11] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT press, 1998.

[12] J. A. Boyan and M. L. Littman, "Packet routing in dynamically changing networks: A reinforcement learning approach," in *Proc. Neural Inf. Process. Syst.* Morgan-Kaufmann, 1994, pp. 671–678.

[13] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Mach. Learn.*, vol. 8, no. 3, pp. 279–292, May 1992.

[14] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.

[15] L. Busoniu, R. Babuska, and B. De Schutter, "Multi-agent reinforcement learning: A survey," in *Proc. 9th Int. Conf. Control, Autom., Robot. and Vis.*, Dec. 2006, pp. 1–6.

[16] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz, "Pruning convolutional neural networks for resource efficient transfer learning," in *Proc. ICLR*, 2017, pp. 1–17.