Toward Packet Routing with Fully-distributed Multi-agent Deep Reinforcement Learning

Xinyu You, Xuanjie Li, Yuedong Xu, Hui Feng
Research Center of Smart Networks and Systems
School of Information Science and Technology
Fudan University, Shanghai, China
{xyyou18, xuanjieli16, ydxu, hfeng}@fudan.edu.cn

Jin Zhao
School of Computer Science
Fudan University, Shanghai, China
jzhao@fudan.edu.cn

***Abstract*—Packet routing is one of the fundamental problems in computer networks in which a router determines the next-hop of each packet in the queue to get it as quickly as possible to its destination. Reinforcement learning has been introduced to design the autonomous packet routing policy namely Q-routing only using local information available to each router. However, the curse of dimensionality of Q-routing prohibits the more comprehensive representation of dynamic network states, thus limiting the potential benefit of reinforcement learning. Inspired by recent success of deep reinforcement learning (DRL), we embed deep neural networks in multi-agent Q-routing. Each router possesses an independent neural network that is trained without communicating with its neighbors and makes decision locally. Two multi-agent DRL-enabled routing algorithms are proposed: one simply replaces Q-table of vanilla Q-routing by a deep neural network, and the other further employs extra information including the past actions and the destinations of non-head of line packets. Our simulation manifests that the direct substitution of Q-table by a deep neural network may not yield minimal delivery delays because the neural network does not learn more from the same input. When more information is utilized, adaptive routing policy can converge and significantly reduce the packet delivery time.**

I. Introduction

Packet routing is a very challenging problem in distributed and autonomous computer networks, especially in wireless networks in the absence of centralized or coordinated service providers. Each router decides to which neighbour it should send his packet in order to minimize the delivery time. The primary feature of packet routing resides in its fine-grained per-packet forwarding policy. No information regarding the network traffic is shared between neighbouring nodes. In contrast, exiting protocols use flooding approaches either to maintain a globally consistent routing table (e.g. DSDV [10]), or to construct an on-demand flow level routing table (e.g. AODV [9]). The packet routing is essential to meet the dynamically changing traffic pattern in today's communication networks. Meanwhile, it symbolizes the difficulty of designing fully distributed forwarding policy that strikes a balance of choosing short paths and less congested paths through learning with local observations.

This work is partially supported by Natural Science Foundation of China (No. 61772139), the National Key Research and Development Program of China (No.213), Shanghai-Hong Kong CollaborativeProject under Grant 18510760900 and CERNET Innovation Project NGII20170209.

Reinforcement learning (RL) is a bio-inspired machine learning approach that acquires knowledge by exploring the interaction with local environment without the need of external supervision [1]. Therefore, it is suitable to address the routing challenge in distributed networks where each node (interchangeable with router) measures the per-hop delivery delays as the reward of its actions and learns the best action accordingly. Authors in [5] proposed the first multi-agent Q-learning approach for packet routing in a generalized network topology.

This straightforward routing policy achieves much smaller mean delivery delay compared with the benchmark shortest path approach. Xia et al. [29] applied dual RL-based Q-routing approach to improve convergence rate of routing in cognitive radio networks. Lin and Schaar [3] adopted the joint Q-routing and power control policy for delay sensitive applications in wireless networks. More applications of RL-based routing algorithms can be found in [11]. Owing to the well-known "curse of dimensionality" [14], the state-action space of RL is usually small such that the existing RL-based routing algorithms cannot take full advantage of the history of network traffic dynamics and cannot explore sufficiently more trajectories before deciding the packet forwarding. The complexity of training RL with large state-action space becomes an obstacle of deploying RL-based packet routing.

The breakthrough of deep reinforcement learning (DRL) provides a new opportunity to a good many RL-based networking applications that are previously perplexed by the prohibitive training burden. With deep neural network (DNN) as a powerful approximator of Q-table, the network designer can leverage its advantages from two aspects: (1) the neural network can take much more information as its inputs, enlarging the state-action space for better policy making; (2) the neutral network can automatically abstract invisible features from high-dimensional input data [17], thus achieving an end-to-end decision making yet alleviating the handcrafted feature selection technique. Recent successful applications include cloud resource allocation [20], adaptive bitrate video streaming [21], cellular scheduling [22]. DRL is even used to generate routing policy in [23] against the dynamic traffic pattern that is hardly predictable. However, authors in [23] considers a centralized routing policy that requires the global topology and

the global traffic demand matrix, and operates at the flow-level. Inspired by the power of DRL and in view of the limitations of Q-routing [5], we aim to make an early attempt to develop fully-distributed packet routing policies using multi-agent deep reinforcement learning.

In this paper, we proposed two multi-agent DRL routing algorithms for fully distributed packet routing. One simply replaces the Q-table of vanilla Q-routing [5] by a carefully designed neural network (Deep Q-routing, DQR). The input information, i.e. the destination of the head of line (HOL) packet in the queue, remains unchanged except for being one-hot encoded. The other introduces extra information as the input of the neural network, consisting of the action history and the destinations of future packets (Deep Q-routing with extra information, DQR-EI). We conjecture that the action history is closely related to the congestion of next hops, the number of packets in the queue indicates the load of the current router, and knowing the destinations of the coming outgoing packets avoids pumping them into the same adjacent routers. With such a large input space, the Q-routing [5] cannot handle the training online and the training of deep neural networks using RL rewards becomes necessary. DQR-EI is fully distributed in the sense that each router is configured with an independent neural network, and it has no knowledge about the queues and the DNN parameters of neighbouring routers. This differs from the recent multi-agent DRL learning framework in other domains [6] where the training of neural networks are simultaneous and globally consistent. The training of multi-agent DRL is usually difficult (e.g. convergence and training speed), while DQR and DQR-EI prove the feasibility of deploying DRL-based packet routing in the dynamic environment.

Our experimental results reveal two interesting observations. Firstly, simply replacing Q-tables by DNNs offers the comparable delivery delay with the original Q-routing. The different representations for the same input implicitly yield almost the same Markov decision process (MDP) policy. Secondly, DQR-EI significantly outperforms DQR and Q-routing in terms of the average delivery delay when the traffic load is high. After examining the routing policy of DQR-EI, we observe that each router makes adaptive routing decision by considering more information than the destination of the HOL packet, thus avoiding congestion on "popular" paths.

The remainder of this paper is organized as follows: Section II reviews the background knowledge of RL and DRL. Section III presents our design of DQR and DQR-EI. The delivery delay of the proposed algorithms is evaluated in Section IV with Q-routing as the benchmark. Section V is devoted to makes discussions about future study and challenges. Section VI concludes this work.

II. Background and Literature

In this section, we briefly review RL and DRL techniques and their applications to routing problem and then put forward the necessity of fully-distributed learning for real-world routing problem.

A. RL algorithm

Based on the mapping relationship between observed state and execution action, RL aims to construct an agent to maximize the expected discounted reward through the interaction with environment. Without prior knowledge of which state the environment would transition to or which actions yield the most reward, the learner must discover the optimal policy by trial-and-error.

The first attempt to apply RL in the packet routing problem is Q-routing algorithm, which is a variant of Q-learning [1]. Since Q-routing is essentially based on multi-agent approach, each node is viewed as an independent agent and endowed with a Q-table to restore Q-values as the estimate of the transmission time between that node and others. With the aim of shortening average packet delivery time, agents will update their Q-table and learn the optimal routing policy through the feedback from their neighboring nodes when receiving the packet sent to them. Despite the superior performance over shortest-path algorithm in dynamic network environment, Q-routing suffers from the inability to fine-tune routing policy under heavy network load and the inadequate adaptability of network load change. To address these problems, other improved algorithms have been proposed such as PQ-routing [7] which uses previous routing memory to predict the traffic trend and DRQ-routing [8] which utilizes the information from both forward and backward exploration to make better decisions.

B. DRL algorithm

DRL embraces the advantage of deep neural networks to the training process, thereby improving the learning speed and the performance of RL [4]. One popular algorithm of DRL is Deep Q-Learning (DQL) [24], which implements a Deep Q-Network (DQN) instead of Q-table to derive an approximate of Q-value with special mechanisms of experience replay and target Q-network.

Recently, network routing problems with different environment and optimization targets are solved with DRL. Based on the control model of the agent, these algorithms can be categorized as follows:

Class 1: Single-agent learning.

Single-agent algorithm treats the network controller as a central agent which can observe the global information of the network and control the packet scheduling strategy of every router. Both the learning and execution process of this kind of algorithm are centralized [25], in other words, the communication between routers are not restricted during training and execution.

With the single-agent algorithm, SDN-Routing [23] presents the first attempt to apply DRL in the routing optimization of traffic engineering. Viewing the traffic demand, which represents the bandwidth request between each source-destination pair, as the environment state, the network controller determines the transmission path of packets to achieve the objective of minimizing the network delay. Another algorithm [19]

considers a similar network model while taking minimum link utilization as the optimization target.

Class 2: Multi-agent learning.

In multi-agent learning, every router in the network is treated as a single agent which can observe only the local environment information and take actions according to its own routing policy.

The first multi-agent DRL learning algorithm applied in the routing problem is DQN-routing [6] by combining Q-routing and DQN. Each router is regarded as an agent whose parameters are shared by each other and updated at the same time during training process (centralized training), but it provides independent instructions for packet transmission (decentralized execution). The comparison with contemporary routing algorithms in online tests confirms a substantial performance gain.

C. Fully-distributed learning

Algorithms with centralized learning process stated above are not applicable in the real computer network. The centralized learning controller is usually unable to gather collected environment transitions from widely distributed routers once an action is executed somewhere and to update the parameters of each neural network simultaneously caused by the limited bandwidth.

Accordingly, for better application in real-world scenario, the routing algorithms we proposed are based on fully-distributed learning, which means both the training process and the execution process are decentralized. Under these circumstances, every agent owns its unique neural network with independent parameters for policy update and decision making, thereby avoiding the necessity for the communications among routers in the process of environment transition collection and parameter update.

III. Design

We establish the mathematical model of the packet routing problem and describe the representation of each element in the reinforcement learning formulation. Then we put forward two different deep neural network architectures substituting the original Q-table and propose the corresponding training algorithm.

A. Mathematical Model

Network. The network is modeled as a directed graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, where $\mathcal{N}$ and $\mathcal{E}$ are defined as finite sets of router nodes and transmission links between them respectively. A simple network topology can be found in Fig. 1, containing five nodes and five pairs of bidirectional links. Every packet is originated from node s and destined for node d: $s, d \in \mathcal{N}$ and $s \neq d$ with randomly generated intervals.

Routing. The mission of packet routing is to transfer each packet to its destination through the relaying of multiple routers. The queue of routers follows the first-in first-out (FIFO) criterion. Each router n constantly delivers the packet in the head of line to its neighbor node v until that packet reaches its termination.

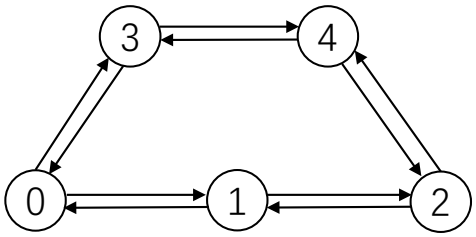


Fig. 1. 5-node network topology.

Target. The packet routing problem aims at finding the optimal transmission path between source and destination nodes based on some routing metric, which, in our experiment, is defined as the average delivery time of packets. Formally, we denote the packet set as $\mathcal{P}$ and the total transmission time as t_p for every packet $p : p \in \mathcal{P}$. Our target is to minimize the average delivery time $T = \sum_{p \in \mathcal{P}} t_p / K$, where K denotes the number of packets in $\mathcal{P}$.

B. Reinforcement Learning Formulation

The packet routing can be modeled as a multi-agent reinforcement learning problem with partially observable Markov decision processes (POMDPs) [28], where each node is an independent agent which can observe the local network state and make its own decisions according to an individual routing policy. Therefore, we will illustrate the definitions of each element in reinforcement learning for a single agent.

State space. The packet p to be sent by agent n is defined as *current packet*. We denote the state space of agent n as $S_n : \{d_p, E_n\}$, where d_p is the destination of the current packet and E_n, which may be empty, is some extra information related to agent n. At different time steps, the state observed by the agent is time varying due to the dynamic change of network traffic.

Action space. The action space of agent n is defined as $A_n : \mathcal{V}_n$, where $\mathcal{V}_n$ is the set of neighbor nodes of node n. Accordingly, for every agent, the size of action space equals to the number of its adjacent nodes, e.g., each node in Fig. 1 has two candidate actions. Once a packet arrives at the head of queue at time step t, agent n observes the current state $s_t \in S_n$ and picks an action $a_t \in A_n$, and then the current packet is delivered to the corresponding neighbor of node n.

Reward. We craft the reward to guide the agent towards effective policy for our target: minimizing the average delivery time. The reward at time step t is set to be the sum of queueing time and transmission time: $r_t = q + l$, where the former q is the time spent in the queue of agent n, and the latter l is referred to as the transmission latency to the next hop.

C. Deep Neural Network

We will introduce two types of algorithms for applying the deep neural network into Q-routing in this section. Essentially, they both replace the original Q-table in Q-routing with a

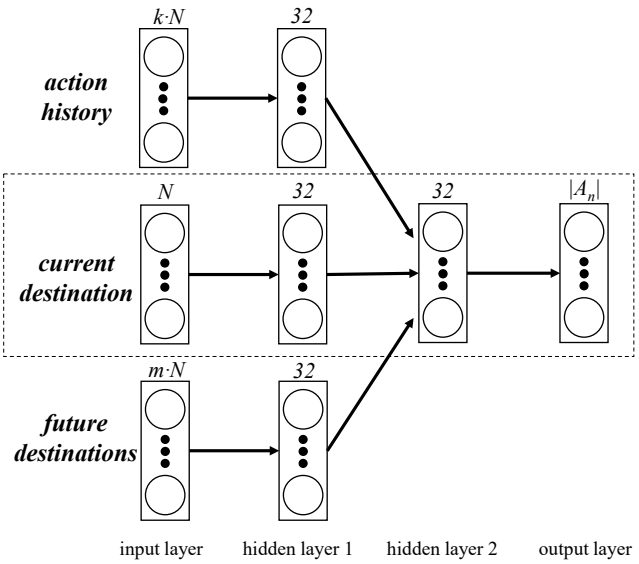


Fig. 2. Fully connected neural network with ReLU activiation.

neural network but utilize different information as the input. Note that in the formulation of reinforcement learning, each node is an individual agent and therefor possesses its own neural network for decision making. Accordingly, the following description of the neural network architecture is tailored for a single agent.

Class 1: Deep Q-routing (DQR)

The primary aim in to find whether there is any improvement if the Q-table, which stores Q-value as a guideline to choose actions, in Q-routing is replaced simply by a neural network without changing the input information. We propose an algorithm called Deep Q-routing (DQR) to compare the different representation of the routing policy.

As shown in the dotted box of Fig. 2, we build a fully connected neural network with two hidden layers and 32 neurons each. The input of the neural network is the one-hot encoding of the current packets destination ID, so that the number of input neurons equals to the total number of nodes in the network topology. For example, for the network with five nodes in Fig. 1, the one-hot encoding result of destination number 4 is [00010]. Furthermore, the size of the output layer and the agents action space $|A_n|$ are identical, and the value of each output neuron is the estimated Q-value of the corresponding action. With this change of the representation for Q-value, we try to update the parameter of neural networks instead of the value of the Q-table.

Class 2: Deep Q-routing with extra information (DQR-EI)

While both DQR and Q-routing would make the constant decision for the packet with the same destination due to the single input, we propose another algorithm called Deep Q-routing with Extra Information (DQR-EI) by integrating more system information into each routing decision.

The input information of the neural network can be classified as three parts: (1) current destination: the destination node of the current packet which is the same as above, (2) action history: the executed actions for the past k packets sent out just before the current packet, (3) future destinations: the destination nodes of the next m packets waiting behind the current packet. Before being input into the neural network, all of such information will be processed with one-hot encoding. As a result of the additional input information, there are some changes in the structure of the neural network. As showed in Fig. 2, the neuron number of the input layer and the first hidden layer is added to hold another two kinds of information, while the second hidden layer and the output layer remain unchanged. With the powerful expression capability of neural networks, the agent of DQR-EI is able to execute adaptive routing policy as the environment of network changes.

In both classes of neural network, we use Rectified Linear Unit (ReLU) as the activation function and Root Mean Square Prop (RMSProp) as the optimization algorithm.

Algorithm 1 Deep Q-routing (with extra information)

// initialization
for agent i = 1, N **do**
  Initialize replay buffer $D_i \leftarrow \varnothing$
  Initialize Q-network Q_i with random weights θ_i
end for

// training process
for episode = 1, M **do**
  for each decision epoch t **do**
    Assign current agent n and packet p
    Observe current state s_t
    Select and execute action

$$a_t = \begin{cases} \text{a random action} & \text{with probability } \epsilon \\ argmax_a Q_n(s_t, a_t; \theta_n) & \text{with probability } 1-\epsilon \end{cases}$$

    Forward p to next agent v_t
    Observe reward r_t and next state s_{t+1}
    Set transmission flag $f_t = \begin{cases} 1 & v_t = d_p \\ 0 & \text{otherwise} \end{cases}$
    Store transition $(s_t, r_t, v_t, s_{t+1}, f_t)$ in D_n
    Sample random batch $(s_j, r_j, v_j, s_{j+1}, f_j)$ from D_n
    Set $y_j = r_j + max_{a'} Q_{v_j}(s_{j+1}, a'; \theta_{v_j})(1 - f_j)$
    $\theta_n \leftarrow$ *GradientDescent* ($(y_j Q_n(s_j, a_j; \theta_n))^2$)
  end for
end for

D. Learning Algorithm

By integrating Q-routing and DQN, we propose the packet routing algorithm with multi-agent deep reinforcement learning, where both training and execution process are set decentralized. The pseudo-code of the learning algorithm is shown in Algorithm 1, in which the initialization and the training process are identical for each node.

Every node i is treated as an individual agent and possesses its own neural network Q_i with particular parameter θ_i to estimate the state-action value function $Q_i(s, a; \theta_i)$, which

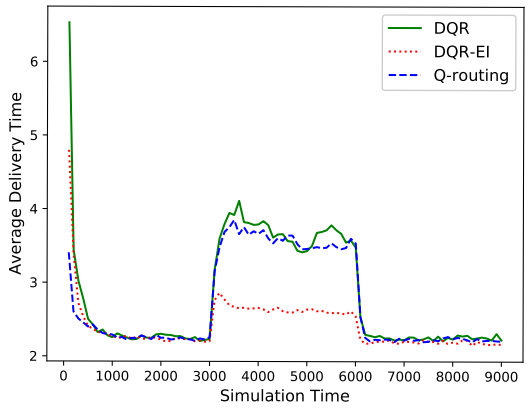


Fig. 3. Online test result.

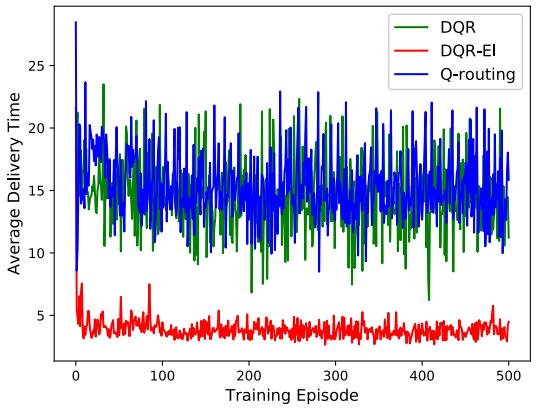


Fig. 4. Offline training speed comparison.

represents the expected delivery time for a packet to reach the destination when the agent executes action a in state s. Replay memory D_i with capacity of 100 is also initialized independently for each agent to restore its environment transitions, and from it a random min-batch with size of 16 will be sampled for the update of its network parameters.

For every decision epoch t when a packet p arrives at the head of line of a certain node n, agent n will observe the current state s_t and execute an action a_t based on ϵ-greedy policy, which means agent n will choose a random action from its action space A_n with probability ϵ or choose the action with the highest Q-value with probability $1-\epsilon$. The assignment of a_t is given by:

$$a_t = \begin{cases} \text{a random action} & \text{with probability } \epsilon \\ argmax_a Q_n(s_t, a_t; \theta_n) & \text{with probability } 1-\epsilon \end{cases} \tag{3.1}$$

Then the current packet p is forwarded to the corresponding neighbor node v_t and the reward r_t is calculated and sent back to agent n. Besides, the transmission flag f_t will be set to 1 if the next node v_t matches the packets destination d_p or set to 0 otherwise. The assignment of f_t is given by:

$$f_t = \begin{cases} 1 & v_t = d_p \\ 0 & \text{otherwise} \end{cases} \tag{3.2}$$

After that, agent n records this transition $(s_t, r_t, v_t, s_{t+1}, f_t)$ into its replay memory D_n and then samples a training batch $(s_j, r_j, v_j, s_{j+1}, f_j)$ randomly from it. As a result of the unstable environment caused by the multi-agent characteristic, the remaining delivery time τ that packet p is expected to spend from v_t to d_p need to be recalculated before the training process. τ is given by:

$$\tau = max_{a'} Q_{v_j}(s_{j+1}, a'; \theta_{v_j}) \tag{3.3}$$

At the end of the decision epoch, the method of gradient descent is used to fit the neural network $Q_n(\theta_n)$. The loss function L is given by:

$$L = (r_j + \tau(1-f_j) - Q_n(s_j, a_j; \theta_n))^2 \tag{3.4}$$

In this way, the network parameters of each agent are updated with episodic training until convergence.

IV. Evaluation

We conducted several experiments in the simulation environment of computer network with different topologies to compare our proposed algorithms DQR and DQR-EI with Q-Routing in both online and offline mode.

A. Simulation environment

Topology. The topology of the computer network we used is the same as Fig. 1, which remains static in the whole experiment. Despite the simple structure, we can explore new insights into packet routing, and actually a more complex network will lead to similar results. All the nodes and links in the network share the same attributes: the buffer size of each node is unlimited and the bandwidth of each link equals to the packet size, in which case only a single packet can be transmitted at a time.

Packet. A certain proportion, named *distribution ratio*, of packets are generated from node 0 (busy ingress-router) to node 2 (busy egress-router), while the other packets source and destination are chosen uniformly. Packets are introduced into the network with the same size and their generated intervals follow Gaussian distribution where a smaller mean value would indicate a higher network load level and the standard deviation is fixed at 0.1 in the whole experiments.

Time setting. The time during the simulation is measured by seconds. The transmission time between adjacent nodes a packet has to spend is set to 1.0 s. The performance criterion of the experiment is the average delivery time of packets within a certain period.

B. Online result

In online simulation environment where packets are generated all the time following the regulations described in Section IV-A, the parameters of neural networks and the value of the Q-table are randomly initialized and are updated from time to time. The simulation timeline is split into intervals of 100s and for every interval the average delivery time of transmitted

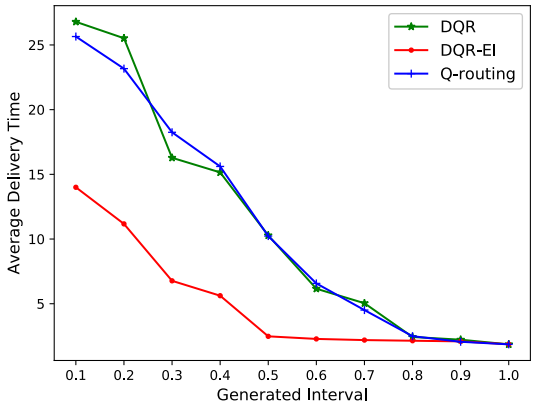


Fig. 5. Offline test with different network loads.

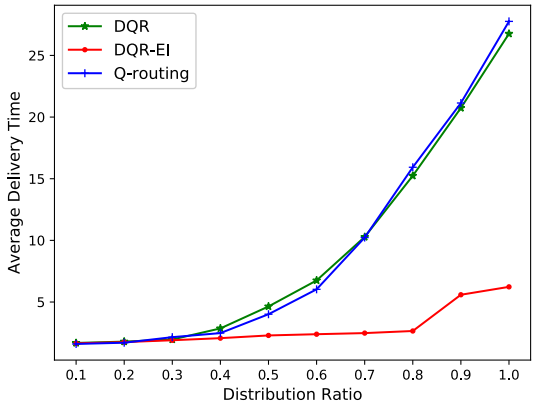


Fig. 6. Offline test with different distribution ratios.

packets is recorded. We initially set the generated interval of packets to 1.0s and suddenly change it to 0.8s at time 3000s and reset it to 1.0s at time 6000s.

Fig. 3 plots the average result of 50 online tests with different source-destination pair sets. We can clearly see that (1) after a short period of adaptation to an unfamiliar environment, all three algorithms find their best performance which remain stable at the same level, (2) when load level is raised at time 3000s, the routing policy of each algorithm begin to change. After the convergence of new policy, the final average delivery time of DQR-EI keeps stable at 2.7s while the others fluctuate around 3.7s. (3) after the reset of load level at time 6000s, the performance of all three algorithms return to their former degree, whereas DQR-EI converges more quickly than DQR and Q-routing, and therefore is more adaptable to dynamic changes of network load.

C. Offline result

In offline experiments, we generated a fixed packet series containing 100 packets as the training set on which we trained the neural network and Q-table individually and saved their convergent models after 500 episodes. Then we restored those well-trained models and compared their performance in an unseen test environment where packets were generated at the corresponding network load level but different source-destination pairs from the training set.

Training speed. With the fixed packet sequence whose generated interval is 0.5s and distribution ratio is 70%, we trained all three algorithms and compared their training speed. Fig. 4 shows the variation trend of average packet delivery time along with the training episode. We find that after about 100 episodes, the average delivery time of DQR-EI keeps stable at a lower level but DQR and Q-routing fluctuates violently from time to time and never converges. At the end of training process, DQR-EI outperforms DQR and Q-routing.

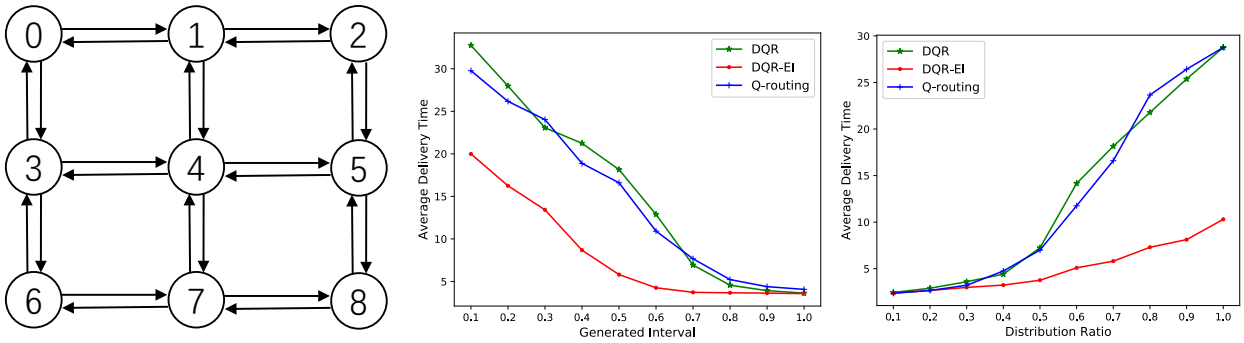
Network load. In terms of the distribution ratio at 70%, the average result of 50 offline tests in various packet generated intervals ranging from 0.1s to 1.0s is depicted in Fig. 5. As expected, we can see that (1) all three algorithms have almost the same performance when the generated interval is between 0.8s and 1.0s (low network load), (2) DQR and Q-routing perform comparably at different network load levels, (3) when the generated interval is between 0.1s and 0.7s (high network load), the average delivery time of DQR-EI is much less than that of DQR and Q-routing, (4) when the generated interval is higher than 0.5s , the average delivery time of DQR-EI remains stable around 2.3s but exceeds 4.5s otherwise in which case link congestion occurs inevitably in unbearable network load.

Distribution ratio. We conducted another 50 offline tests with various distribution ratio ranging from 10% to 100% when the generated interval is fixed at 0.5s. The average result is shown in Fig. 6. Similarly, we can find that (1) all three algorithms perform equally well when the distribution ratio is between 10% and 20% in which case the spatial distribution of packet generation is approximately uniform), (2) DQR and Q-routing have comparable performance at different distribution ratios, (3) the performance gap between DQR-EI and DQR as well as Q-routing increases with the rise of the distribution ratio, (4) the average delivery time of DQR-EI keeps stable around 2.3s at the initial stage, but when the distribution ratio is higher than 80%, it exceeds 4.5s unexpectedly due to insufficient bandwidth caused by massive packets transmission requests from node 0 to node 2.

D. Complex topology

In the above experiments, each router in the 5-node topology (Fig. 1) has two neighbour nodes to choose during the packet delivery process. To test the scalability of our proposed routing algorithm, we expand the network scale to a 3×3 topology depicted in Fig. 7(a) where the connection of routers becomes complex and each router has more choices to make, thus increasing the difficulty of decision making. The attributes of the new topology are the same as those in Section IV-A except that node 0 is viewed as the busy ingress-router and node 8 is viewed as the busy egress-router.

Similar to the experiment setting in Section IV-C, we execute simulation experiments in two cases: (1) fixing the distribution ratio and changing the network load level with different packet generated intervals; (2) fixing the generated intervals and changing the the distribution ratio. The simula-



(a) 3×3 topology (b) Different network loads (c) Different distribution ratios

Fig. 7. Offline test result in 3×3 topology

tion results are shown in Fig. 7(b) and Fig. 7(c) respectively. We can see that, in 3×3 topology, the variation trends of the average delivery time with respect to the network load level and distribution ratio have fairly consistency with those in the 5-node topology. The similar results in different topologies demonstrate good robustness of our proposed algorithms.

E. Performance analysis

From the findings in online and offline tests with different topologies, we can draw a conclusion that DQR can achieve the same performance as Q-routing while DQR-EI outperforms both of them. We will clarify the main reasons for this result from the perspective of the routing policy learned by each algorithm. To simplify our interpretation, the following analysis is based on the 5-node topology (Fig. 1).

Neural network and Q-table. After the algorithm converges, a certain agent of DQR executes almost the same routing strategy as that of Q-routing provided that the destination of the current packet to be transmitted is identical. As described in Section III-C, the only difference between the two algorithms is the representation of Q-value while the learning algorithm does not change. The neural network which is merely the approximation of Q-table would not help with the estimate of the accurate transmission time between the source and termination of packets. As a result, DQR and Q-routing who share the same input have comparable performance under different conditions.

Additional information. Whenever the agents of DQR and Q-routing choose an action, the only information they can utilize is the destination of the current packet, leading to the same routing decision for packets with the same destination. For this reason, the sole input will cause violent fluctuations during the process of offline training (Fig. 4). For example, when transmitting a packet destined for node 2, the agent of node 0 has no idea which neighbour node should be sent to because of the alternate congestion in link 0→1 and link 0→3. However, as described in Section III-C, the input layer of the neural network of DQR-EI contains additional information besides that stated above: the action history of previous packets and destinations of future packets, with which the agents of DQR-EI can execute different but effective routing policy for every packet despite the same destination.

Adaptive routing policy. More precisely, we evaluate the case where the generated interval and distribution ratio are set to 0.1s and 70% respectively in the offline test. In this simulation environment, the network load is so heavy that a large number of packets are waiting in the queue of node 0 to be transmitted to node 2 in every unit time. For these packets, the agent of node 0 has two choices: traveling them through node 1 or through node 3 and 4. The well-trained agent of node 0 of DQR and Q-routing follows the shortest path and therefore all those packets will be sent to node 1. Under this strategy, serious congestion will occur unavoidably in the link 0→1 and link 1→2, which eventually lead to longer delivery time. However, DQR-EI can overcome this difficulty cleverly. Before making decisions for every packet destined for node 2 at node 0, the agent will collect the information about the actions the last five packets have taken and the nodes the next five packets are destined for. For example, when the last five packets were sent to node 1, the agent decides to send the current packet to node 3 regardless of the longer path to avoid long latency. Similarly, after some packets were sent to node 3, the agent will change its policy and decide to transfer the packet through node 1 again. Therefor, with additional information, DQR-EI has the ability to grasp the dynamic changes in the network and adjust its routing policy accordingly, which, shown in our test result, can gain shorter average delivery time and a better performance.

V. Discussion

In this section, we put forward our research plan and ensuing challenges in several directions, deriving from some limitations of the current work.

Other DRL algorithms. The routing algorithms we proposed (DQR and DQR-EI) are based on DQN [24], which is a classical but simple form of DRL. Thanks to the tremendous contribution researchers in the society of DRL have made, more effective DRL algorithms can be leveraged in packet routing. For example, as the optimization of the policy gradient based RL algorithm, TRPO [12] is combined with the neural

network in continuous control domain to ensure monotonic performance [13]. Besides, based on DPG [15], an off-policy actor-critic algorithm, DDPG [16] uses the neural network as a differentiable function approximator to estimate action-value function, and then updates the policy parameters in the direction of the deterministic policy gradient.

Realistic simulation environment. In this paper, the experiments we have conducted in the simulation environment are based on some restrictive conditions, as described in Section IV-A, which would impede the adoption of our proposed routing algorithms in the realistic computer network with complex traffic patterns. In the future work, we will consider a more general network setting such as routers with finite queue and packets with varied sizes. Furthermore, NS-3 network simulator [2] can be utilized as the standard platform to test the performance of routing algorithms.

Multi-agent Learning. The packet routing system in our current implementation is built on the multi-agent model where each router is treated as an independent agent and learns asynchronously without communication. However, in multi-agent environment, the inherent non-stationarity problem [18] during the learning process is magnified after the application of DNN. To address this problem, importance sampling and fingerprint [26] provides alternative solutions. Moreover, witnessing the benefits of cooperative learning [27], we will analyse the performance boost that the local coordination of DNNs (e.g., parameter sharing and memory sharing [25]) can yield in our future study.

VI. Conclusion

We presented two fully-distributed packet routing algorithms, DQR and DQR-EI, based on multi-agent deep reinforcement learning. From the preliminary experiment results, we find that the implementation of neural networks, serving as the substitute for Q-table, in DQR cannot get better estimate of Q-value for the same input, thereby resulting in comparable performance with Q-routing. After introducing additional system information into neural networks in DQR-EI, agents can learn adaptive routing policy in the dynamically changing environment and the average packet delivery time can be reduced to a considerable extent.

References

[1] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[2] T. R. Henderson, M. Lacage, G. F. Riley, C. Dowell, and J. Kopena, Network simulations with the ns-3 simulator, *SIGCOMM demonstration*, vol. 14, no. 14, p. 527, 2008.

[3] Z. Lin and M. van der Schaar, Autonomic and distributed joint routing and power control for delay-sensitive applications in multi-hop wireless networks, *IEEE Transactions on Wireless Communications*, vol. 10, no. 1, pp. 102113, 2011.

[4] N. C. Luong *et al*., Applications of Deep Reinforcement Learning in Communications and Networking: A Survey, *arXiv preprint arXiv:1810.07862*, 2018.

[5] J. A. Boyan and M. L. Littman, Packet routing in dynamically changing networks: A reinforcement learning approach, in *Advances in neural information processing systems*, 1994, pp. 671678.

[6] D. Mukhutdinov, A. Filchenkov, A. Shalyto, and V. Vyatkin, Multi-agent deep learning for simultaneous optimization for time and energy in distributed routing system, *Future Generation Computer Systems*, vol. 94, pp. 587600, 2019.

[7] S. P. Choi and D.-Y. Yeung, Predictive Q-routing: A memory-based reinforcement learning approach to adaptive traffic control, in *Advances in Neural Information Processing Systems*, 1996, pp. 945951.

[8] S. Kumar and R. Miikkulainen, Dual reinforcement Q-routing: An on-line adaptive routing algorithm, in *Proceedings of the artificial neural networks in engineering Conference*, 1997, pp. 231238.

[9] C. Perkins, E. Belding-Royer, and S. Das, Ad hoc on-demand distance vector (AODV) routing, 2003.

[10] C. E. Perkins and P. Bhagwat, Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers, in *ACM SIGCOMM computer communication review*, 1994, vol. 24, pp. 234244.

[11] H. A. Al-Rawi, M. A. Ng, and K.-L. A. Yau, Application of reinforcement learning to routing in distributed wireless networks: a review, *Artificial Intelligence Review*, vol. 43, no. 3, pp. 381416, 2015.

[12] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, Trust region policy optimization, in *International Conference on Machine Learning*, 2015, pp. 18891897.

[13] Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel, Benchmarking deep reinforcement learning for continuous control, in *International Conference on Machine Learning*, 2016, pp. 13291338.

[14] Y. Bengio, A. Courville, and P. Vincent, Representation learning: A review and new perspectives, *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 17981828, 2013.

[15] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, Deterministic policy gradient algorithms, in *ICML*, 2014.

[16] T. P. Lillicrap et al., Continuous control with deep reinforcement learning, *arXiv preprint arXiv:1509.02971*, 2015.

[17] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, A brief survey of deep reinforcement learning, *arXiv preprint arXiv:1708.05866*, 2017.

[18] P. Hernandez-Leal, M. Kaisers, T. Baarslag, and E. M. de Cote, A survey of learning in multiagent environments: Dealing with non-stationarity, *arXiv preprint arXiv:1707.09183*, 2017.

[19] A. Valadarsky, M. Schapira, D. Shahaf, and A. Tamar, Learning to route with deep rl, in *NIPS Deep Reinforcement Learning Symposium*, 2017.

[20] H. Mao, M. Alizadeh, I. Menache, and S. Kandula, Resource management with deep reinforcement learning, in *Proceedings of the 15th ACM Workshop on Hot Topics in Networks*, 2016, pp. 5056.

[21] H. Mao, R. Netravali, and M. Alizadeh, Neural adaptive video streaming with pensieve, in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, 2017, pp. 197210.

[22] Z. Xu, Y. Wang, J. Tang, J. Wang, and M. C. Gursoy, A deep reinforcement learning based framework for power-efficient resource allocation in cloud RANs, in *2017 IEEE International Conference on Communications (ICC)*, 2017, pp. 16.

[23] G. Stampa, M. Arias, D. Sanchez-Charles, V. Munts-Mulero, and A. Cabellos, A deep-reinforcement learning approach for software-defined networking routing optimization, *arXiv preprint arXiv:1709.07080*, 2017.

[24] V. Mnih et al., Human-level control through deep reinforcement learning, *Nature*, vol. 518, no. 7540, p. 529, 2015.

[25] J. Foerster, I. A. Assael, N. de Freitas, and S. Whiteson, Learning to communicate with deep multi-agent reinforcement learning, in *Advances in Neural Information Processing Systems*, 2016, pp. 21372145.

[26] J. Foerster et al., Stabilising experience replay for deep multi-agent reinforcement learning, in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, 2017, pp. 11461155.

[27] J. Foerster, I. A. Assael, N. de Freitas, and S. Whiteson, Learning to communicate with deep multi-agent reinforcement learning, in *Advances in Neural Information Processing Systems*, 2016, pp. 21372145.

[28] G. E. Monahan, State of the arta survey of partially observable Markov decision processes: theory, models, and algorithms, *Management Science*, vol. 28, no. 1, pp. 116, 1982.

[29] B. Xia, M. H. Wahab, Y. Yang, Z. Fan, and M. Sooriyabandara, Reinforcement learning based spectrum-aware routing in multi-hop cognitive radio networks, in *2009 4th International Conference on Cognitive Radio Oriented Wireless Networks and Communications*, 2009, pp. 15.