A Deep Reinforcement Learning-based Routing Scheme with Two Modes for Dynamic Networks

Peizhuang Cong[†], ★Yuchao Zhang[†], Wendong Wang[†], Ke Xu[*], Ruidong Li[‡], Fuliang Li[✳]
[†]Beijing University of Posts and Telecommunications, [*]Tsinghua University
[‡]National Institute of Information and Communications Technology, [✳]Northeastern University

***Abstract*—With the development of communication and transmission technologies, more and more applications, like Internet of vehicles and tele-medicine, become more sensitive to network latency and accuracy, which requires routing schemes to be more efficient. In order to meet such urgent need, learning-based routing strategies emerges, with the advantages of high flexibility and accuracy. These strategies can be divided into two categories, centralized and distributed, enjoying the advantages of high precision and high efficiency, respectively. However, routing become more complex in dynamic network, where the link connections and access states are time-varying, so these learning-based routing mechanisms are required to be able to adapt to network changes in real time. In this paper, we designed and implemented both two of centralized and distributed reinforcement learning-based routing schemes (RLR-T). By conducting a series of experiments, we deeply analyzed the results and gave the conclusion that the centralized is better to cope with dynamic networks due to its faster reconvergence, while the distributed is better to handle with large-scale networks by its high scalability.**

***Index Terms*—routing, deep reinforcement learning, DQN**

I. Introduction

Along with the burgeoning development of the Internet in recent years, emerging network applications, such as industrial Internet, IoV (Internet of Vehicles), 4K/8K video transmission and edge computing, are requiring routing schemes to be more efficient. Nevertheless, today's network is no longer as stable as the traditional wired network due to the access of a large number of mobile devices, which make network connection status change frequently. The conflict between application requirement and network characteristic brings great challenges to the network in providing efficient and flexible routing decisions.

To ensure the QoS (Quality of Service) in current dynamic networks, improving hardware infrastructures not only causes huge cost but also has limitations of performance improvement. Meanwhile, a research of CAIDA [1] has shown that the existing network still has a lot of room for optimization. However, many traditional mathematical model-based optimization schemes [2], [3], which aim to simplify specific scenarios through idealized assumptions to solve network optimization problems, cannot be guaranteed due to uncertainties in the real scene. Then, the ML(Machine Learning) brings new ideas to solve this problem.

Given the development of ML and SDN (Software Defined Network), ML-based intelligent routing scheme has noble feasibility. Data-driven intelligent routing has features of high accuracy and extreme versatility. Models trained by different data set can solve various network optimization problems without complicated network environment assumptions and modeling. Some existing researches show that the ML-based routing strategies have been successfully applied in many scenarios, such as opportunistic networks [4], wireless networks [5], IoT [6], and improved in accuracy and performance than the traditional routing protocol [6]. The deployment of the ML-based routing scheme can be divided into two categories, centralized and distributed. The centralized gets the globe network state and makes globe routing decisions via a centralized controller which is akin to the SDN controller, and the distributed makes single routing hop by each router. Most of current networks are time-varying, whose connections be established and canceled at any time, or nodes be accessed and eliminate. Both the centralized and distributed require the model to achieve convergence or it will make wrong decision. The aforementioned dynamics cause the fitted model non-convergence again.

In this paper, we designed and implemented both centralized and distributed (based on Deep Reinforcement Learning method) routing schemes, then compared and analyzed the performance of the two models, especially under dynamic conditions, through a series of experiments. The results show that the centralized scheme is suitable to deal with dynamic networks due to its faster reconvergence, and the distributed scheme is better to handle with large-scale networks by its high scalability.

II. Background and Related Work

Along with the development of the Internet, more and more devices, such as smartphones, automobiles, which result in quite dynamic networks [7]. The time-varying connections of links and access states in a dynamic network require routing strategies to vary to adapt the network. ML-based decision system can fit the optimal result only when the model converges. The dynamic mentioned above will exercise a great influence on it, which needs re-train the model to re-fit the current network status. In the rest of this section, some

★Corresponding author: Yuchao Zhang (yczhang@bupt.edu.cn).
This work is supported in part by the National Key Research and Development Program of China (2019YFB1802603), The National Natural Science Foundation of China (NSFC) Youth Science Foundation (61802024), and The Fundamental Research Funds for The Central Universities under Grant (2482020RC36). Ruidong Li's work was partly supported by JSPS KAKENHI Grant Number JP19H04105.

representative works on reinforcement learning and AI-based network optimization schemes are presented.

A. AI-based Network Optimizations

Given the successful application of machine learning in natural language processing, computer vision, and other fields, many scholars try to apply AI technology to network context, including congestion control, resource allocation, security, and so on [8]–[11]. Zhang et al. optimized the video caching strategy of edge servers through model prediction [12]. Liang et al. optimized the decision tree generation strategy for flow classification by an reinforcement learning model [13]. Hua et al. put forward a reward-clipping mechanism to stabilize GAN-DDQN training against the effects of widely-spanning utility values to solve the problem of several slices in a radio access network with base stations which share the same physical resources [14]. In the realm of routing, Mao et al. proposed a DBN(Deep Belief Network)-based routing scheme in the backbone of the network, where the domain's border routers calculate the best inter-domain path for packets [15]. Xiao et al. combined a deep learning model and the link reversal theory to generate a DDAG (Decision Directed Acyclic Graph) and assign some weights to all links, then choosing an optimal path via a greedy algorithm when making routing decisions [16].

B. Reinforcement Learning

Reinforcement learning learns the optimal strategy by maximizing accumulated rewards which is suitable for decision-making problems [17]–[19]. The agent chooses the action A_t according to the strategy π in the current state S_t. The environment transfers to the next state S_{t+1} according to A_t. The agent receives the fed back reward R_t by the environment and chooses the next action according to strategy π. DRL combines the advantages of deep neural networks and traditional reinforcement learning, such as DQN, SARSA, etc., to solve the problem of excessive data dimensions in the Qtable-based RL model. Following some classic DRL models, like A3C, DDPG, and MADDPG, were proposed.

Xu et al. used DRL for intra-domain traffic engineering optimization, and proposed the DRL-based traffic engineering solution, DRL-TE, which uses traditional methods to generate paths and uses a DRL unit to adjust the routing path split ratio online [20]. Valadarsky et al. tried to predict the future network traffic based on historical traffic data through the DRL unit and calculated the appropriate routing configuration based on the predicted results [21]. Ramy et al. developed a hierarchical cluster-oriented adaptive per-flow path calculation mechanism by leveraging the DDQN(Deep Double Q-Network) algorithm, where the end-to-end paths are calculated by the source nodes with the assistance of cluster leaders at different hierarchical levels [22].

III. Models Design

In this Section, we propose a RLR-T with two modes, centralized and distributed, whose details are presented.

TABLE I
Declaration of Notations

Notation	Declaration
N	the set of all nodes
N_i	means router i, and $N_i \in N$
E	the set of all links
E_{ij}	means the link of N_i to N_j, if it is exist then $E_{ij} \in E$, otherwise, $E_{ij} \notin E$
des	destination of a task
S	state of the model environment
A	action
R	reward of state change, $R_{S_i \to S_j}$ means reward of change from S_i to S_j. R_{max} means the biggest prize when the task is done and R_{min} means the harshest punishment which usually a very small negative number
C_{ij}	the cost of transform from N_i to N_j, such time consumption or decrease of utilization.
q_target	a network trained by data of state change and related reward
q_value	a network evaluates the action and updated by q_target

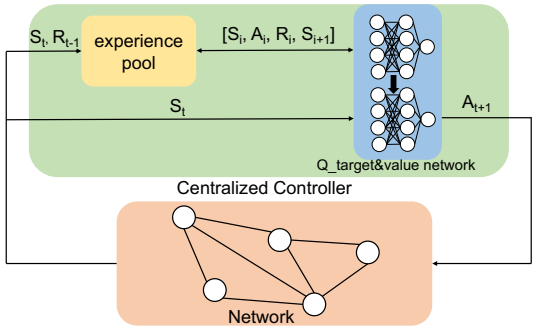


Fig. 1. Centralized RL-based routing model.

A. Variable Definition

A network topology is defined as a directed graph, ¡N, E¿. All routers are defined as a node set N, and router i is N_i. And all links are denoted as edge set E, $E_{ij} \in E$ means there is a direct link between N_i and N_j. All declarations are shown in TABLE I.

B. Centralized Routing Scheme

1) Overall Structure: This fashion needs a central controller that akin to the SDN controller to make all routing decisions. It can train the DRL-based model offline and push forwarding rules to all routers via OpenFlow protocol. It is different from the traditional routing protocol in this mode. All routers no longer need to interact with neighbors, but the state of the entire network is maintained by the controller. The centralized mode can be directly upgraded in the existing network running the OpenFlow protocol and does not require to do much work on the underlying equipment.

As shown in the Fig. 1, the environment generates a reward based on state change after performing the last action. The reward is usually related to the optimal criteria, if the transmission target is to minimize delay, the smaller the delay of routing decision, the greater the reward, and vice versa. In a period, it involves the state before action, action, state after

the action, reward, and whether the task is done. This five-tuple will be stored in the experience pool of RL to make the previous five-tuples used for training the q_target network independently. The state after action will be taken as the input of q_value which will output an action with biggest reward of this state. The environment performs this action and feedbacks a reward, then a new period is completed. The pseudocode of the centralized process is shown in the Algorithm 1. The q_target obtains previous data from the experience pool and trains its model. The q_value has the same network structure as q_target, and its parameters are updated periodically by the q_target.

Algorithm 1 Centralized Process.

Initial: $S_{current}$ = environment.state()

```
# Interaction Process:
while TRUE do
    A = q_value(S_current)
    S_next, R = environment.execute(A)
    experience_pool.save([S_current, A, R, S_next])
    S_current = S_next
end while
# Training Process:
while TRUE do
    replay_data = experience_pool.sample(batch_size)
    q_target.train(replay_data)
    if ifupdate == TRUE then
        q_value.copy_parameters(q_target)
    end if
end while
```

2) Detailed Design: If routing decisions on all transport tasks need to be made using a model, then there must be at least two essential properties for a packet forwarding, the source address, and the destination address. Besides, because forwarding satisfies a Markov random process, the address of the router that is processing the packet is the source address in the current state, which is independent of all previous states. Therefore, we define that the state has two attributes, the current and the destination, as shown in the Eq(1). After performing the forwarding action, only the current will be modified, and the destination address remains to be consistent with the entire transmission process.

$$S = [N_{current}, N_{des}] \tag{1}$$

The reward is a significant part of DRL, which needs to be designed with different calculation rules according to diverse tasks and the transition of environmental states. Before that, it was explained that the action space is nodes set N, and the action output by the model is N_i, $\in N$, such as Eq(2).

$$A = [N_{next}] \tag{2}$$

First, if N_{next} is not directly connected, the reward is the harshest punishment, R_{min}, which is usually a very small negative number. Second, if $N_{current}$ is N_{next}, the reward is the maximum reward, R_{max}, which is usually a large positive

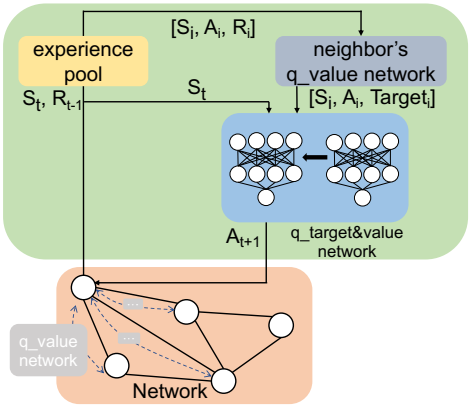


Fig. 2. Distributed RL-based routing model.

number. For the third case, N_{next} is not N_{des} and it can be reached directly by $N_{current}$, the reward is a positive coefficientthe α multiply the minimum cost of from N_{next}'s neighbors to N_{des} minus the cost of from N_{next} to N_{des}. As shown in formula Eq(3). Due to the iterative computation, the value of $N_{current} = N_{next} = N_{des}$ is required, which is already included in the above case because the cost of from one to itself is 0.

$$R_{[i,des]\to[j,des]} = \begin{cases} R_{min}, & E_{ij} \notin E \\ R_{max}, & E_{ij} \in E \ \&\ j = des \\ \alpha(argmin_{k\in neighbor(i)}(C(k,des)) - C(j,des), & E_{ij} \in E \ \&\ j \neq des) \end{cases} \tag{3}$$

Since the controller has the global perspective of the network, it can calculate the cumulative reward value for all nodes that made forwarding decisions along the routing path based on historical optimal routing decision and the final routing decision performance. Then, it can put the reward value as the label to train the q_target, as shown in the Eq4.

$$q\_target_{[i,des]\to[j,des]} = R_{[i,des]\to[j,des]} \tag{4}$$

The q_value network is periodically updated by q_target using the copy scheme.

C. Distributed Routing Scheme

1) Overall structure: Different from the above-mentioned centralized mode, the distributed mode does not require a centralized controller. Each router only needs to maintain its link states to its neighbors and interact q_value network with them. It is similar to the traditional routing protocol, but the optimal path is calculated through the RL model. And the information in the routing table is no longer the distance metrics but some parameters of q_value network.

Each router in the network needs to maintain its model. Take one router as an example, as shown in the Fig. 2. The router

Algorithm 2 Distributed Process.

Take N_i as example:
Initial: $S_{current}$ = get_des(packet)

```
# Interaction Process:
while TRUE do
    A = q_value(S_current)
    R = environment.execute(A)
    experience_pool.save([S_current, A, R])
    S_current = get_des(packet)
end while
# Training Process:
while TRUE do
    replay_data = experience_pool.sample(batch_size)
    q_target_label = replay_data + cumulative_reward()
    q_target.train(replay_data)
    if ifupdate == TRUE then
        q_value.copy_parameters(q_target)
    end if
end while
# Diffusion and Reception of q_value:
while TRUE do
    if ischange(q_value) == TRUE then
        send_neighbors(q_value)
    end if
    if isreceive(q_value) == TRUE then
        N_i.update_neighbors(q_value)
    end if
end while
```

quantifies environmental states and rewards and stores them in the experience pool. When training the q_target network, it is necessary to combine the q_value network of the maintained neighbors and the reward to calculate the target value. This process takes into account the cumulative reward to prevent the decision model from only selecting the optimal link based on the immediate reward. The parameters of q_value are maintained and updated by q_target. Taking the transmission task label as the input of q_value and selecting the maximum value action according to the calculation result. This action will be sent to the router forwarding module or as an entry updated to the local routing table. When the action result of q_value changes, the latest network parameters need to be diffused to neighbors. The pseudocode of the distributed process is shown in the Algorithm 2.

2) Detail Design: Since the distributed mode does not have a global perspective and all routers only maintain their state and neighbor q_values, considering that the forwarding of data packets satisfies the Markov random process, the state in this mode only needs to have a destination parameter, such as the Eq(5) shown. The action of the model is the same as the centralized mode as Eq(2).

$$S = [N_{des}] \tag{5}$$

The reward calculation of distributed mode can be divided into the following two situations. The first, if N_{next} is not N_{des}, whether it can be reached directly by $N_{current}$ or not, the reward is the harshest punishment, R_{min}. The second is that the N_{next} can be reached directly by $N_{current}$ and it is N_{des}, then the reward is the maximum reward, R_{max}, minus the cost of from $N_{current}$ to N_{next} multiply the positive coefficient, α. Again, it includes the value of $N_{current}$ = N_{next} = N_{des} due to the cost of from one to itself. As shown in the Eq(6).

$$R_{\to j}^{i,[des]} = \begin{cases} R_{min}, & E_{ij} \notin E \text{ or } j \neq des \\ R_{max} - \alpha C_{ij}, & E_{ij} \in E \ \& \ j = des \end{cases} \tag{6}$$

To prevent the model from causing local optimum by choosing actions based only on a one-step reward, the q_target network update takes into account two parts. One is the reward immediately obtained by the action, and the other is the long-term cumulative reward, which is calculated by neighbor's q_value network according to the action. The formula is as Eq(7), which means that the target value from $[N_i]$ to $[N_j]$ under destination node is N_{des} should be the direct reward from $[N_i]$ to $[N_j]$ plus the max value calculated by N_j's q_value to $[N_{des}]$. This is an iterative process, and it will finally converge to a stable state.

$$q\_target_{\to j}^{i,[des]} = R_{\to j}^{i,[des]} + \gamma max(q\_value_{\to k}^{j,[des]} | N_k \in N) \tag{7}$$

In the same way, the q_value is updated by q_target periodically as mentioned above.

IV. Evaluation

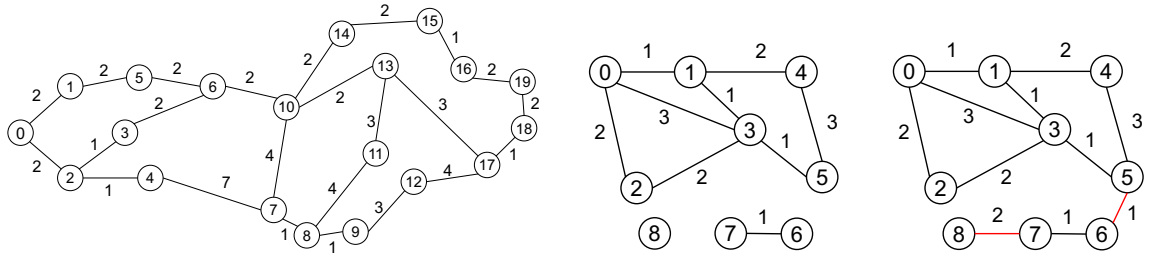
In this section, we explain the experimental setup and analyzed the RLR-T in convergence time and reconvergence time when network state changes under different network scales.

A. Experiment Setup

The two modes of centralized and distributed RL-based models are implemented by TensorFlow 1.3.0 and on the Ubuntu 16.04-LTS operating system. The GPU is GeForce GTX 970 and the CPU is Intel Xeon 3.30GHz × 8.

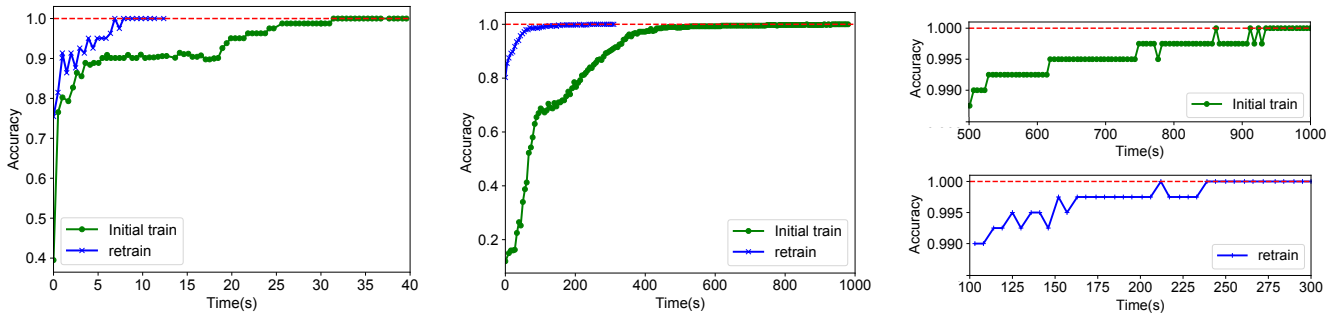
Topologies: Two topologies, with 20 nodes and 8 nodes, were set up in the experiment shown in the Fig. 3(a) and Fig. 3(b). The 20-nodes one is based on Savvis 2011 USA topology [23]. To increase the number of alternative paths in this topology, three routers are added to form some loops. Link weights (delays) are assigned. And setting acnodes(N_8) and unconnected subgraphs(N_6 and N_7) in 8-nodes one is to verify the validity of setting edge weights to represent node exits. The model training data is transmission task set generated randomly in the corresponding topology.

In the process of model retraining, topology changes as following rules, the 20-nodes changes the weight of some links randomly, and 8-nodes is connected to all subgraphs, as shown in the Fig. 3(c).



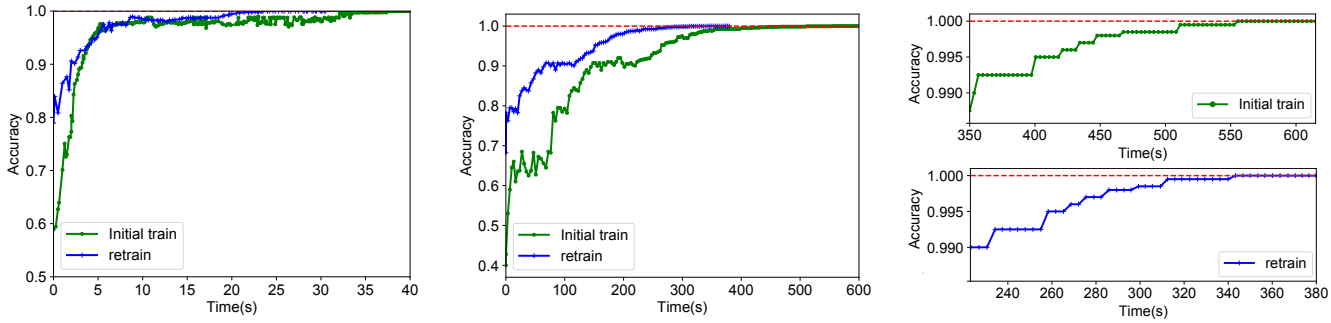
(a) 20-N (b) 8-N (c) 8-N'

Fig. 3. Topologies



(a) 8-N topology (b) 20-N topology (c) Tail accuracy of 20-N

Fig. 4. Centralized mode.



(a) 8-N topology (b) 20-N topology (c) Tail accuracy of 20-N

Fig. 5. Distributed mode.

B. Experimental Results

Based on the experimental setup above, we use centralized and distributed strategies respectively in the two topologies. The verification label of accuracy is the globally optimal decision calculated according to the shortest path algorithm. The accuracy of each calculation is to verify whether the forwarding port of all points to other points except itself conforms to the label. It needs to be explained that this paper only puts forward a common model to compare the situation under the two modes horizontally, and does not make model and decision optimization for the requirements under special scenarios, which is the future work.

Centralized mode*:* In this mode, the relationship between accuracy and training time in the two topologies is shown in the Fig.4(a) and 4(b). The retraining process is to save the convergent model of the initial training first, and then modify the network state, such as modifying links weight randomly or adding or deleting nodes, and then load the saved model parameters for retraining. On the basis of some fitting ability, the accuracy of the model at the beginning time has reached a relatively high, so it can reconverge faster than the initial training. In the 8-N topology, initial training time, as the line marked by '•' in Fig.4(a), takes about 30 seconds to stabilize at 100% accuracy. The retraining time, as the line marked by '|' in Fig.4(b), only takes 10 seconds to reach 100% accuracy state. In the 20-N topology, the accuracy goes up quickly, but it also takes a long time to reach 100% accuracy. Similarly, the time of retraining is significantly shorter than its the initial training.

Distributed mode: In this mode, each point needs to process distribute training first, which means the model should fit neighbors states, and then completes the federal training of the entire network via the neighbor-to-neighbor q_value exchange. In the 8-N topology, due to the small scale, the neighborhood fitting process makes it achieve relatively high accuracy quickly, reaching 90% accuracy in less than 5 seconds and 100% in less than 40 seconds, as shown in the Fig.5(a). The time to reach 90% accuracy of retraining is similar to the initial training, but the time to reach 100% is slightly shorter. In the 20-N topology, the fluctuation in the beginning stage of initial training is severe, because the updating of neighbor q_value has a great influence on the decision making, and the global q_value convergence is a conjunct process. The early stage of retraining is more stable, and the convergence state with high accuracy can be achieved more quickly, as shown in the Fig.5(b).

From excellent to perfect: For the model, it is time-consuming to converge to 100% accuracy. The tail accuracy performance from 99% to 100% in 20-N topology as shown in Fig.4(c)and 5(c). Tail convergence accounts for a large proportion of the overall situation. We analyze the decision results of the tail stage. The most are suffered fitting fluctuation between the sub-optimal and the optimal. Only a tiny learning rate and multiple iterations can complete the final fitting. And it will be more obvious when the sub-optimal and the optimal are close, but this is not necessary. One is that it's acceptable to choose a sub-optimal path approaching optimal when there is only one optimal criterion. The second is that multi-optimal criteria are usually considered in the current network, such as selecting the comprehensive optimal path subjected by delay and bandwidth, so using top N optimal sets intersection to relax the model fitting accuracy, and then avoiding the tail convergence time.

Analysis: According to the above results, the distributed mode requires a process akin to route switching, so the reconvergence time is not as good as the centralized mode. However, when the scale gradually increases, the distributed can improve accuracy and converge faster than the centralized. On large scale, directional diffusion optimization can further improve the performance of the distributed mode, but the centralized mode will be greatly affected.

V. Conclusion

Although machine learning can break through some bottlenecks of traditional methods in the network, it still has some limitations, such as only applicable to some special scenarios. This paper designed the centralized and distributed RL-based routing decisions, compared and analyzed time of model convergence and reconvergence, and pointed out the advantages and characteristics of each mode in different networks. The future work will design the most suitable RL-based routing decision model for diverse network scenarios and transmission requirements based on the conclusion.

References

[1] CAIDA, "The cooperative associate for internet data analysis," http://www.caida.org/data, 2020.

[2] S. Ghorbani, Z. Yang, P. B. Godfrey, Y. Ganjali, and A. Firoozshahian, "Drill: Micro load balancing for low-latency data center networks," in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, 2017, pp. 225–238.

[3] P. Kumar, Y. Yuan, C. Yu, N. Foster, R. Kleinberg, P. Lapukhov, C. L. Lim, and R. Soulé, "Semi-oblivious traffic engineering: The road not taken," in *15th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 18)*, 2018, pp. 157–170.

[4] S. K. Dhurandher, J. Singh, M. S. Obaidat, I. Woungang, S. Srivastava, and J. J. Rodrigues, "Reinforcement learning-based routing protocol for opportunistic networks," in *ICC 2020-2020 IEEE International Conference on Communications (ICC)*. IEEE, 2020, pp. 1–6.

[5] X. Guo, H. Lin, Z. Li, and M. Peng, "Deep reinforcement learning based qos-aware secure routing for sdn-iot," *IEEE Internet of Things Journal*, 2019.

[6] Z. Zhuang, J. Wang, Q. Qi, H. Sun, and J. Liao, "Toward greater intelligence in route planning: A graph-aware deep learning approach," *IEEE Systems Journal*, 2019.

[7] P. Cong, Y. Zhang, W. Wang, and B. Bai, "Dnd: The controllability of dynamic temporal network in smart transportations," in *2019 IEEE Globecom Workshops (GC Wkshps)*. IEEE, 2019, pp. 1–6.

[8] K. Zhang, Y. Zhu, S. Leng, Y. He, S. Maharjan, and Y. Zhang, "Deep learning empowered task offloading for mobile edge computing in urban informatics," *IEEE Internet of Things Journal*, pp. 7635–7647, 2019.

[9] K. Xiong, S. Leng, X. Chen, C. Huang, C. Yuen, and Y. L. Guan, "Communication and computing resource optimization for connected autonomous driving," *IEEE Transactions on Vehicular Technology*, pp. 12 652–12 663, 2020.

[10] K. Xiong, S. Leng, C. Huang, C. Yuen, and Y. L. Guan, "Intelligent task offloading for heterogeneous v2x communications," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–13, 2020.

[11] B. Yang, X. Cao, K. Xiong, C. Yuen, Y. L. Guan, S. Leng, L. Qian, and Z. Han, "Edge intelligence for autonomous driving in 6g wireless system: Design challenges and solutions," *arXiv preprint:2012.06992*, 2020.

[12] Y. Zhang, P. Li, Z. Zhang, B. Bai, G. Zhang, W. Wang, B. Lian, and K. Xu, "Autosight: Distributed edge caching in short video network," *IEEE Network*, pp. 194–199, 2020.

[13] E. Liang, H. Zhu, X. Jin, and I. Stoica, "Neural packet classification," in *Proceedings of the ACM Special Interest Group on Data Communication*, 2019, pp. 256–269.

[14] Y. Hua, R. Li, Z. Zhao, X. Chen, and H. Zhang, "Gan-powered deep distributional reinforcement learning for resource management in network slicing," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 2, pp. 334–349, 2019.

[15] B. Mao, Z. M. Fadlullah, F. Tang, N. Kato, O. Akashi, T. Inoue, and K. Mizutani, "Routing or computing? the paradigm shift towards intelligent computer network packet transmission based on deep learning," *IEEE Transactions on Computers*, vol. 66, no. 11, pp. 1946–1960, 2017.

[16] S. Xiao, H. Mao, B. Wu, W. Liu, and F. Li, "Neural packet routing," in *Proceedings of the Workshop on Network Meets AI & ML*, 2020, pp. 28–34.

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

[18] Y. Dai, K. Zhang, S. Maharjan, and Y. Zhang, "Deep reinforcement learning for stochastic computation offloading in digital twin networks," *IEEE Transactions on Industrial Informatics*, pp. 1–1, 2020.

[19] Y. Zhang, K. Zhang, J. Cao, H. Liu, and S. Maharjan, "Deep reinforcement learning for social-aware edge computing and caching in urban informatics," *IEEE Transactions on Industrial Informatics*, 2019.

[20] Y. Zhan, P. Li, and S. Guo, "Experience-driven computational resource allocation of federated learning by deep reinforcement learning," in *Proc. of IPDPS*, 2020.

[21] A. Valadarsky, M. Schapira, D. Shahaf, and A. Tamar, "Learning to route," in *Proceedings of the 16th ACM workshop on hot topics in networks*, 2017, pp. 185–191.

[22] R. E. Ali, B. Erman, E. Baştuğ, and B. Cilli, "Hierarchical deep double q-routing," in *ICC 2020-2020 IEEE International Conference on Communications (ICC)*. IEEE, 2020, pp. 1–7.

[23] Zoo, "The internet topology zoo," http://www.topology-zoo.org/, 2020.