Research on Routing Algorithm Based on Reinforcement Learning in SDN

Chuntao Fang, Cheng Cheng, Zhongyuan Tang and Chuanhuang Li

School of Information and Electronic Engineering, Zhejiang Gongshang University, Zhejiang Hangzhou 310000, China

E-mail: 1147197814@qq.com

Abstract. Reinforcement learning is an unsupervised learning method which has been used in many fields. Actually, the essence of Reinforcement learning is a decision-making problem. It constantly tries to interact with the environment. Each interaction process will get a different feedback value, and then it adjusts each trial strategy through feedback. In this paper, we apply the Reinforcement learning technology to software defined network routing algorithm, and propose the routing algorithm based on Q-learning. Through the combination of Reinforcement learning and neural network, which means the Q-table in Q-learning is replaced by neural network, we present routing algorithm based on Deep Q-learning.

1. Introduction

In recent years, mobile Internet and Internet of Things are rapidly developing. New applications and fast-growing data traffic require better quality of service (QoS). Software defined network (SDN)[1][2][3] separates the infrastructure layer and the control layer of the network device to realize the intelligent management of network traffic. Usually, SDN controller[4][5][6] forwarding the packet bases on Dijkstra algorithm. If forwarding of all packets only relies on the shortest path algorithm, it may cause a serious problem that all data streams are brought together by selecting the same forwarding path, which greatly reduces the utilization of network link and leads to network congestion. Therefore, we need a better strategy to manage the routing of the SDN network[7] At present, there is no fixed routing algorithm applied in SDN network routing. Designing a routing algorithm under the SDN network and finding a good forwarding path for the data packet is a hot spot in the current SDN network research.

Hoanghhuu V [8] proposed a dynamic routing algorithm based on the maximum correlation pair model to find a path from the starting point to the ending point in the unknown environment. Compared with the traditional dynamic routing algorithm, the experimental results show that the algorithm has higher search efficiency. Jie Sun et al. [9] proposed the SAS (Scheduling According to Stickness) algorithm. The SAS algorithm implements the QoS requirements of the link based on the link sticking value. Generally, the data stream with smaller link sticking value will enjoy the right to forward preferentially. The importance of the data flow is measured by the link sticking value. So the data stream with the higher priority is forwarded first, and the data stream with the lower priority is forwarded later. Huaidong Zhang et al. [10] modeled the path selection of elephant flow based on multi-commodity flow problems. Compared with the ECMP (Equal Cost Multi Path) [11] algorithm, network bandwidth can be fully utilized to improve network utilization.
Mentioned above, SDN network does not consider more about the requirements of link’s quality of service. Therefore, ensuring service deployment and optimal utilization of network links in the SDN network architecture, using certain QoS requirements as the criterion for judging the path pros and cons, reinforcement learning is applied to SDN network routing in this paper. Reinforcement learning can continuously explore the surrounding environment through agents without prior knowledge, familiar with the entire environment after a number of training cycles, and make the right choice finally. Different from the Dijkstra algorithm [12] using in traditional network routing, we propose and implement a routing algorithm based on Q-learning. Also, because of the feature that routing algorithm Q-learning needs the defects of artificial design, using the neural network to replace the Q-table in the traditional network, the routing algorithm based on Deep Q-learning is proposed.

2. Model research

2.1. Research on routing algorithm based on Q-learning

Q-learning as a no-model advanced version of reinforcement learning, relying on Q-value iteration to achieve the balance with exploration and utilization. During the iterative process of the Q-learning algorithm, people generally select the direction in which Q-value is the largest. However, not every iteration will follow this principle. Under the results of multiple experiments, Q-learning algorithm will use a certain probability of which the distribution will be carried out in the direction with the largest possible that Q-value is the largest. There are several common terms in Q-learning, states, reward, and actions. The goal of Q-learning is to achieve the maximum reward value by Q-value iteration.

When designing the routing algorithm, a numerical iterative solution method is used to approximate the optimal value. The complex network environment in which new packets are located is a finite state Markov process. When a new packet selects an action from the action set, the network environment accepts the action and transitions state, giving the decision a reward value at the same time. The greater the reward value that a packet gets from an action, then the probability that the packet will select this action again will increase. At a certain time \( t \), the state of the packet is \( s_t \), the data packet selects an action \( a_t \), the network environment state is transferred from \( s_t \) to \( s_{t+1} \), and the reward value given by the environment system is \( r_t \), then the routing decision process in the environment is shown in equation (1).

\[
r_t = r(s_t, a_t, s_{t+1})
\]

In the learning process, the optimal value function is approximated by iteratively calculating \( Q(s, a) \). The basic transfer rules of Q-learning algorithm are shown in equation (2) and equation (3). \( s, a \) respectively represent the current state and action of the packet, \( s' \) and \( a' \) respectively represent the next state and action of the current state, and the learning parameter \( \gamma \) is a constant satisfying \( 0 > \gamma > 1 \).

If the learning parameter tends to 0, indicating that packet routing decision cares more about the reward value of the current action. If the learning parameter tends to 1, packet routing decision will take into account the reward value of the future action.

\[
Q(s, a) = R(s, a) + \gamma \max_{a'} \{Q(s', a')\}
\]

(2)

\[
Q(s, a) = (1 - \lambda_t)Q(s, a) + \lambda_t \{r + \gamma \max_{a' \in A} Q(s', a')\}
\]

(3)

Without supervision, the packet only study relying on the current experience, and it constantly moves from one state to another to explore until it reaches the target state. Each exploration of a packet is called an episode. When the packet arrives at the target state from the initial state, it represents the end of an episode.

The basic steps based on the Q-learning routing algorithm can be summarized as follows:

1) Give the learning parameter \( \gamma \) and the bonus value matrix \( R \).
(2) When the Q-matrix is initialized, set the value of all elements in the matrix to 0.

(3) For each episode:
   (3.1) Select a state $s$ randomly from state set as an initial state.
   (3.2) Before reaching the target state, the following steps will be performed:
      (3.2.1) Select an action $a$ among all possible actions of the current state $s$.
      (3.2.2) Obtain the next state $s'$ with the selected action $a$.
      (3.2.3) Calculate $Q(s, a)$ according to the algorithm transfer rule mentioned above.
      (3.2.4) Let $s = s'$.

Once the Q-matrix training is completed, the newly transmitted data packet can easily find a forwarding path from the arbitrary state $s_0$ to the target state from the trained Q-matrix. The specific process is as follows:

(1) Let the current state $s = s_0$.
(2) Determine the action $a$ in the current state, which satisfies $Q(s, a) = \max \{Q(s, a')\}$.
(3) Let the current state $s = s'$ ($s'$ denotes the next state corresponding to $a$).
(4) Repeat (2) and (3) until $s$ becomes the target state.

The iterative process for every route attempt using of packet which is based on the Q-learning routing algorithm is shown in figure 1. $S_0$ represents the position of the start state of the packet in one iteration process, and $S_n$ represents the state of one iteration. The direction indicated by the arrow is the direction in which the packet state transitions. It can be seen that based on the Q-learning routing algorithm, each state transition of the packet occurs between two adjacent states, so such a Q-learning algorithm is also called a single-step Q-learning algorithm. The single-step Q-learning algorithm takes up a relatively large amount of memory and at the same time converges slowly.

\[
\text{Step one training} \quad \text{Step two training} \quad \text{Step three training} \quad \ldots \quad \text{Step n training}
\]

\[
\begin{align*}
S_0 \quad & S_0 \quad S_0 \quad S_0 \\
\uparrow \quad & \uparrow \quad \uparrow \quad \uparrow \\
S_1 \quad & S_1 \quad S_1 \quad S_1 \\
\uparrow \quad & \uparrow \quad \uparrow \quad \uparrow \\
S_2 \quad & S_2 \quad S_2 \quad S_2 \\
\uparrow \quad & \uparrow \quad \uparrow \quad \uparrow \\
S_1 \quad & S_3 \quad S_4
\end{align*}
\]

**Figure 1.** Iterative process based on Q-learning routing algorithm

**Figure 2.** Algorithm test simulation experiment environment model diagram

### 2.2. Research on routing algorithm based on deep Q-learning

The Q-learning algorithm can learn from the surrounding environment, but it needs to manually design the corresponding features to train a convergent Q-matrix, while neural networks have good processing power for massive data. So replaced the Q-matrix in Q-learning with a neural network, it is the Deep Q-learning algorithm. A Deep Q-learning algorithm model is a combination of a multi-layer neural network model and a reinforcement learning model. Give a state $s$, output a vector of action values $Q(s, \ast; \theta)$, $\theta$ is the parameter of the network. And neural network as a function can map the vector from an n-dimensional space to an m-dimensional action space. The Deep Q-learning algorithm has two main points, one is the target network, and the other is experience replay. Equation (4) is the target network update equation.
\[ Y_t^{\text{DQN}} = R_{t+1} + \gamma \max_a Q(S_{t+1}, a; \Theta^*) \]  

(4)

For experience replay in reinforcement learning, the observed state transitions are first stored. After the samples have accumulated to a certain extent, then sampling randomly to update the network. Experience replay is a very important part of deep reinforcement learning, which can greatly improve the system performance of deep reinforcement learning.

There are two key elements based on the Deep Q-learning routing algorithm: one is the calculation of the loss function, and the other is the extraction of the training samples in the neural network training process.

Calculation of the loss function: The relationship between the target Q-value and the old Q-value in Q-learning coincides with the relationship between the output value and the result value in the neural network learning, so the loss function is calculated as shown in equation (5).

\[ \text{loss} = (\text{target}_q - q)^2 \]  

(5)

Training sample extraction: Based on the concept of experience replay, the reason for randomly sampling is that a sample obtained after a packet exploring by the surrounding environment is a time-associated sequence with temporal correlation. Because of the time correlation, if the data is directly used to train and update the Q-table, the system convergence will be greatly affected. Therefore, we adopt random sampling to solve the problem of time correlation.

The complete process based on the Deep Q-learning routing algorithm is as follows:

1. Initialize packet randomly as one state, initialize the memory pool, and set the observation value.
2. Loop traversal:
   1. Packet policy selects an action \( a \).
   2. After select an action \( a \). Obtain the corresponding bonus value \( r \), the state \( s' \) after the action ends, and the value done which judges whether the entire search ends.
   3. Save the relevant parameters \( s, a, r, s', \text{done} \) in the first two steps to the memory pool.
   4. Determine the amount of data stored in the memory pool (whether the number of data in the memory pool exceeds the set observation value). If not, go to step (2.5). If the data is sufficient, perform the following steps:
      1. Take a part of the data randomly from the memory pool as samples for training.
      2. Take sample state \( s' \) which is taken randomly as training samples, and obtain the Q-table corresponding to state through training.
      3. Calculate \( \text{target}_q \) value table corresponding to Q-table according to equation (6).
      4. Train the neural network by \( q \) and \( \text{target}_q \).
4. Determine if the search process is over.
   1. Find the end, set a state to \( s \) randomly.
   2. If the search is not completed, the current state \( s \) is updated to \( s' \).

3. Experimental simulation design

3.1. Routing algorithm simulation environment

The tests of the two algorithm models are based on the network topology shown in figure 2. In this experiment, the link is graded by a certain QoS standard. The higher the level, the better the link QoS, and the stronger the trend of selecting the link. The ultimate goal of the experiment is to select a forwarding path with better QoS performance for the packet.
3.2. Routing Algorithm Status Description

The main points of reinforcement learning are to find out the state of the environmental system and the corresponding executable actions. According to the simulation experiment environment model diagram above, all the state sets under the adopted topology can be described as follows: \( S = [S_1, S_2, S_3 \ldots S_{15}, L^*_1, L^*_2, L^*_3]. \) \( S_1 \sim S_{15} \) respectively represents 15 switches in the experimental environment, * placeholder represents the links of any two switches. In order to fully represent all the possible link conditions in the entire network environment, each link performance is divided into three levels according to the QoS standard: \(_1\), \(_2\), \(_3\). The link performance corresponding to each level is shown in Table 1. The higher the percentage of link QoS performance, the better the link status. Since there are two or more links on a switch, each link is divided into three different levels, so one link of a switch usually corresponds to an actual state. All states can be materialized according to three different levels of link state. Take switch S3 as an example. There are three links connected to it: L2, L5, and L8. So the three links correspond to three states, and each link can be divided into three states according to the link performance level. The three links of a switch actually correspond to nine specific states. As shown in Table 2.

Table 1. Link performance corresponding to different link levels

| Link level | Link QoS performance |
|------------|----------------------|
| \( L^*_1 \) | 0% ~ 30%             |
| \( L^*_2 \) | 31% ~ 60%            |
| \( L^*_3 \) | 61% ~ 100%           |

S1 = [3, L2_1] represents the state s1, the switch in which it is located is S3, the link is L2, and the link level is \(_1\). After all the states are determined, the optional set of actions in the experimental environment is determined. An optional action refers to moving from one state to another. In this paper, the optional state is defined as \( A = \{s[i], s[t]\} \), which means to transfer from one state \( i \) to another state \( t \).

3.3. Routing Algorithm Reward Function Definition

The purpose of the routing algorithm is to select a link with better comprehensive performance from among the many optional links. In this paper, the network QoS standard is taken as the main consideration. So the purpose of the routing algorithm is to select a path with better QoS synthesis. Since the ultimate goal of reinforcement learning algorithm we adopted is to select the link with the larger reward value as the better link after training, the setting of the bonus value \( R \) is particularly critical.

Since the performance of the link has been divided into three levels, the reward value can be designed based on the performance level of the link. Generally speaking, we will choose the link with higher link performance value as the better link, so according to the three levels of link performance, the set bonus value is as follows: \{[link QoS performance: 0%~30% Reward value \( R_1 = 50 \)], [Link QoS performance: 31% to 60% Reward value \( R_2 = 100 \)], [Link QoS performance: 61% to 100% Reward value \( R_3 = 150 \)]\}.

3.4. Deep reinforcement learning algorithm parameter setting

Deep Q-learning uses neural network to replace the Q-table in Q-learning. In this paper, we design a simple three-layer neural network. After setting the parameter debugging and operating, if the loss function value continues to decrease, as shown in figure 3, confirm that the neural network can be normal operation. The loss function, which can also be called the error function, represents the closeness of the sample's predicted value to the actual value. Learning rate, the number of layers, and the number of neurons in neural network can be adjusted according to the experimental results. In the
experiment, after debugging the learning rate several times, find that setting the value to 0.01 is appropriate.

![Figure 3](image1.png)  **Figure 3.** Neural network loss function value changes with training period

![Figure 4](image2.png)  **Figure 4.** Routing loop problem solving strategy flow chart

3.5. **Routing Loop Problem**

In the above experimental scheme design, there may be a routing loop problem. And traditional RIP protocol has four main methods to solve the routing loop problem [13]: trigger update, horizontal split, toxicity reversal and suppression timing method. We adopt a kind of horizontal splitting method: if a packet passes through a switch node, set a variable Node = [S1, S2, S3,...Sn] to record the switch that the packet has already experienced. The optional path of the next hop of the packet can only be selected from the remaining switch nodes, but cannot be returned to the node that has already arrived. The strategy flow chart for solving the loop problem is shown in figure 4.

4. **Experimental results and analysis**

4.1. **Routing Algorithm Experimental Results**

| Table 3. Q-matrix based on Q-learning training |
|------------------------------------------------|
| 0 0 0 0 252 252 0 0 0 0 0 0 0 0 0 0 0 |
| 0 0 0 0 201 0 0 0 0 0 0 0 0 0 0 0 0 0 |
| 0 0 0 0 0 0 252 0 0 0 0 0 0 0 0 0 0 0 |
| 0 160 0 0 252 0 0 252 0 0 0 0 0 0 0 0 0 |
| 201 0 0 201 0 252 0 0 316 316 0 0 0 0 0 0 |
| 201 0 0 201 0 252 0 252 0 0 316 316 0 0 0 0 0 |
| 0 0 201 0 0 252 0 0 0 0 0 0 316 0 0 0 0 |
| 0 0 0 201 0 0 0 0 0 0 0 0 316 0 0 0 0 |
| 0 0 0 0 252 0 0 0 0 0 0 0 0 0 0 0 0 0 |
| 0 0 0 0 252 252 0 0 0 0 0 0 0 0 0 0 0 396 |
| 0 0 0 0 252 0 0 0 0 0 0 0 0 0 0 0 0 396 |
| 0 0 0 0 0 0 252 0 0 0 0 0 0 0 0 396 0 |
| 0 0 0 0 0 0 252 0 0 0 0 0 0 0 0 0 0 396 0 |
| 0 0 0 0 0 0 252 0 0 0 0 0 0 0 0 0 396 0 |
The final result of the Q-learning algorithm is to train a convergent Q-matrix, through which the link from a given starting point to the end point can be directly selected. The results of the Q-matrix solved by the Q-learning algorithm in this experiment are shown in Table 3. Through the trained Q-matrix, it is easy to get the full path results from any starting point to the ending point 15 as shown in Table 4. Based on the same topology design, the Deep Q-learning routing algorithm can also easily find the complete path from any starting point to the ending point 15. The partial search results after the model training is completed are shown in Table 5.

### Table 4. Arbitrary start-to-end routing results

| Start | Path         | Start | Path         |
|-------|--------------|-------|--------------|
| 0     | 0->4->8->13->15 | 1     | 1->3->4->8->13->15 |
| 2     | 2->6->11->14->15 | 3     | 3->4->8->13->15  |
| 4     | 4->8->13->15   | 5     | 5->9->13->15   |
| 6     | 6->11->14->15  | 7     | 7->12->13->15  |
| 8     | 8->13->15      | 9     | 9->13->15      |
| 10    | 10->14->15     | 11    | 11->14->15     |
| 12    | 12->13->15     | 13    | 13->15         |
| 14    | 14->15         |       |               |

### 4.2. Analysis of experimental results of routing algorithm

Based on the given network topology environment, the number of steps required to reach the end point in the training process based on the Q-learning routing algorithm is analysed. The results of partial cycle convergence steps are shown in table 6. It can be easily seen from figure 5 that based on the Q-learning-based routing algorithm, the convergence speed near the end state does not change the convergence speed of the current state.

### Table 5. Partial routing results of deep Q-learning routing algorithm

| Start | Path         |
|-------|--------------|
| 0     | 0->4->8->13->15 |
| 1     | 1->3->4->8->13->15 |
| 2     | 2->6->11->14->15 |
| 3     | 3->4->8->13->15 |

### Table 6. Q-learning convergence steps

| Cycle | 10 | 20 | 30 | 40 | 50 | 60 | 70 |
|-------|----|----|----|----|----|----|----|
| QL    | 104| 73 | 26 | 9  | 7  | 8  | 6  |

In terms of algorithm accuracy, the Q-learning belongs to unsupervised learning. Each training period in the early stage is in the process of trying and training the Q-matrix, so the accuracy rate is very low, but as the number of training increases, the Q-matrix gradually converges. During the process, it can be found that the accuracy of training is rising rapidly as shown in Table 7. And it is
easy to see from figure 6 that the accuracy of the Q-learning can reach a stable value faster. The environment used in this training process is not complicated, and the final accuracy of the algorithm can reach almost 100%.

**Table 7.** Q-learning accuracy rate

| Cycle | 10  | 20  | 30  | 40  | 50  | 60  | 70  | 80  | 90  | 100 | 110 |
|-------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| QL    | 0.72| 0.88| 0.985| 0.9957| 0.9984| 0.9999| 0.9998| 1   | 1   | 1   | 1   |

**Figure 5.** the average number of steps to the end point changes with the training period

**Figure 6.** Accuracy rate as a function of training cycle

5. Conclusion

In this work, we mainly use reinforcement learning to SDN network routing, and design a routing algorithm based on Q-learning. In the design process of this algorithm, the network link QoS standard is mainly considered. Obviously, this algorithm can make full use of the network link. From the experimental test results, the algorithm can find the route with better QoS performance with nearly 100% accuracy after a certain period of training. There is a disadvantage in the Q-learning-based routing algorithm: the corresponding features need to be manually designed. However, in the actual application environment system, the features are often much complicated. It is very difficult to extract the design features manually, and the effect is not satisfactory. Therefore, we combine neural network with reinforcement learning, replace the Q-table with an approximate function trained by neural networks, and proposes a routing algorithm based on deep reinforcement learning. The experimental results show that the routing algorithm based on Deep Q-learning also shows good performance.

Acknowledgments

This work was supported in part by a grant from the National Key Research and Development Program of China (No.2017YFB0803202), the Natural Science Foundation of China(61871468), the Zhejiang Provincial Natural Science Foundation of China (LY18F010006), the Zhejiang's Key Project of Research and Development Plan (2017C03058), the Zhejiang Provincial Key Laboratory of New Network Standards and Technologies (No.2013E10012), Scientific Research Projects of Zhejiang Education Department (Y201737967).

References

[1] Foundation O N 2012 Software-defined networking The New Norm for Networks.
[2] Shunmiao Z and Fumin Z 2013 Survey on software defined network research Application Research of Computers 30 pp 2246-51.
[3] Hongyi Z, Peyman K, George V and Nick M 2014 Automatic test packet generation IEEE/ACM Transactions on Networking 22 pp 554-566.
[4] Dixit A, Hao F, Mukherjee S, Lakshman T V and Kompella R 2013 Towards an elastic distributed SDN controller Acm Special Interest Group on Data Communication 43 pp 7-12.
[5] Uhlig R, Neiger G and Rodgers D 2005 Intel virtualization technology *IEEE Computer* **38** pp 48-56.

[6] Bosshart P, Gibb G, Kim H S, Varghese G, McKeown N, Izzard M, et al 2013 Forwarding metamorphosis: fast programmable match-action processing in hardware for SDN *ACM SIGCOMM Computer Communication Review* **43** pp 99-110.

[7] Yingya G, Zhiliang W, Xia Y, Xingang S and Jianping W 2014 Traffic engineering in SDN/OSPF hybrid network 2014 IEEE 22nd International Conference on Network Protocols, Raleigh NC USA pp 563-568.

[8] Hoanghuu V, Sanghyeok A and Taechoong C 2011 Extended dyna-q algorithm for path planning of mobile robots *Journal of Measurement Science and Instrumentation* **2** pp 283-287.

[9] Jie S, Li L and Subin S 2016 A routing strategy based on qos and dynamic load balancing *Computer Technology and Development* **26** pp 188-194.

[10] Huaidong Z and Qingwei D 2016 A method of data center network dynamic traffic scheduling based on SDN *Computer and Modernization* pp 80-86.

[11] Hopps C E 2000 Analysis of an equal-cost multi-path algorithm *Journal of Allergy & Clinical Immunology* **109** S265.

[12] Bauer R, Delling D, Sanders P, Schieferdecker D, Schultes D and Wagner D 2008 Combining hierarchical and goal-directed speed-up techniques for dijkstra’s algorithm *WEA’08 Proceedings of the 7th international conference on Experimental Algorithms* pp 303-318.

[13] Fan Y 2012 RIP routing protocol analysis and configuration brief *Silicon Valley* pp 16-17.