# MSc Dissertation

# First Report

# MSc in Telecommunications

**Student:** Jinhao Wu

**Student number: 18110353**

**Project Title: Routing with Reinforcement learning**

University College London

Dept. of Electrical and Electronic Engineering

2019

**Introduction**

Machine learning is very popular and in demand in recent years. It is an interdisciplinary subject and it is also the core technology of artificial intelligence. The machine learning is currently developing in many fields of science and industry. The types of the machine learning differ in the type of their input and output or the type of tasks they are intended to complete. Currently, the machine learning is mainly divided into the following three categories: supervised learning, unsupervised learning, and reinforcement learning [1]. The supervised learning uses training data with labels (real values) to train the model, and the training model is fitted to the real model by some iterative updating method [1]. The supervised learning is mainly applied in classification problem and regression problem. The unsupervised learning uses the unlabelled training data to train the model. Through the analysis of the characteristics of the training data, the feature of the distribution of the dataset can be found [1]. The unsupervised learning is mainly applied in dimension reduction problem and clustering problem. The reinforcement learning does not use training datasets to train the model, but the model itself interacts with the environment under a certain policy and get the response from the environment, and ultimately it finds the most appropriate action which can get maximum benefit [1]. The reinforcement learning is mainly applied in controlling problem and dynamic programming problem.

The reinforcement learning is distinguished from the other two types of machine learning methods by its high biological relevance and high learning autonomy [2]. In reinforcement learning, the six most important components are agent, environment, state, policy, action, observation and reward. The agent is the core of the reinforcement learning. It makes an action according to the policy to interact with the environment in which the agent is located and then it transfers to a new state in the environment. Then the environment returns the reward of the action made by the agent, the agent updates the policy according to the reward obtained and makes new action until the agent moves to the termination state. The reinforcement learning process is shown in the figure 1.

Agent

Environment

Observation

Action

Reward

Figure 1: The reinforcement learning process

In recent years, as people's demand for network scale, performance, and diversification has gradually increased, networks have become more complex and variable in topology, channel, and mobile modes. Therefore, network routing has become a thorny issue. Traditional routing algorithms use pre-defined policies or rules, and each node processes a policy only for a certain network condition or state [3]. Since the network state or condition may change rapidly, these policies may not be optimal in other network states, so these polices may not get the best routing performance some of the time due to the unpredictability of the network [3].

The machine learning is very suitable for solving such problem due to its high adaptability and high performance. The applications of the machine learning, especially the reinforcement learning in complex and changeable network routing problems, have received a considerable attention in scientific and industrial fields [3]. The reinforcement learning can give network nodes the ability of context awareness and intelligence [3]. The context awareness allows network nodes to observe their local network environment parameters, including packet arrival rate, delay, jitter, queuing, congestion, and so on [3]. The Intelligence allows network nodes to have the ability to find the best routing policy, which usually varies with the changes of its network parameters [3]. In other words, the context awareness and the intelligence enable network nodes to make the best forwarding actions based on the observed network parameters and also give the network nodes the ability to learning to get the optimal or near-optimal routing policy, such as the minimum end-to-end delay policy, the minimum packet loss rate policy and the optimal load balancing policy.

**Problem Statement**

This project aims to build a virtual network environment in python and realize a routing algorithm in this network which based on the reinforcement learning. The specific requirements are shown as follows. Firstly, the virtual network environment should contain several routers, which are connected to each other to form a network topology. Secondly, these nodes should be initialized with different network parameters, such as different port bandwidth, different buffer size, different packet processing time, different packet arrival rate, and so on. Thirdly, the forwarding behaviour of these nodes should also be defined, including reading packets from the buffer, processing routing information in the packets, finding the optimal forwarding port, and forwarding along the optimal port to the next hop node. Fourthly, the response behaviour of the network environment according to the node forwarding behaviour should also be defined, including determining whether the packet is lost, whether the packet times out, and whether the packet is routed. Fifthly, a network traffic model should also be defined, this traffic model should have both authenticity and simulation friendliness. For routing algorithms, the specific requirements are shown as follows. Firstly, the algorithm should be based on the reinforcement learning. The node forwards the packet according to the current routing policy and gets a reward from the environment, so that the routing policy is continuously optimized according to the rewards and finally the optimal routing performance is achieved. Furthermore, the routing policy of these network nodes should be presented in the form of neural network rather than traditional tabular routing table. Therefore, this project should realize a reinforcement learning algorithm combined with the deep learning, namely deep reinforcement learning algorithm. This project should also define a statistical and plotting module that records the routing information of each packet forwarded in the network, and finally plots the routing performance figure of each node, which can make the observation of the routing performance become more intuitive. Finally, for the performance of the routing algorithm, the reinforcement learning algorithm should meet or exceed the traditional routing algorithms such as OSPF and RIP.

**Background and literature Review**

**Machine learning in Network routing**

Applying machine learning in network routing is not a not a novel concept. The concept of machine learning techniques in network routing has been proposed more than 20 years ago. In the past 20 years, many network engineers and scientists have proposed their network routing algorithms based on the machine learning, such as reinforcement learning algorithms, swarm intelligence algorithms, genetic algorithms, neural network algorithms, etc. [3]. In 2007, Forster [4] compared these machine learning routing algorithms and stated that the routing algorithm based on the reinforcement learning is more suitable for small scale and energy-constrained networks than other algorithms due to it uses smaller routing overhead. Therefore, this project studied and researched a routing algorithm based on the reinforcement learning.

**Reinforcement learning fundamental**

**Model based reinforcement learning**

At present, most of the reinforcement learning algorithms are basically based on the Markov Decision Process (MDP), which describes the stochastic policy and possible feedback reward that the agent can achieve in an environment with Markov property. The MDP was Proposed by the Russian mathematician Andrei Markov, which is an important theoretical basis for the reinforcement learning [5]. An MDP mainly consists of two parts, namely, the agent and the environment. The agent interacts with the environment by making an action and the environment responds to the agent by returning a reward. In general, an MDP generally consists of five components {S, A, T, R, P}, where S is the set of all states in the environment, A is the set of all actions of the agent, T is the state transferring probability matrix of the agent, R is the reward matrix of the environment, and P is the policy function of the agent, whose input is the current state of the agent and the output is the action made by the agent [5]. It can be seen from the five components of the MDP that the actions made by the agent in the environment are only related to the current state and not related to the previous historical state, this property is called the Markov property, and making decisions and receiving returns in the Markov environment is called Markov Decision Process []. The network routing process is an MDP, because when the node receives the packet, its forwarding behaviour is only based on its own routing table and not related to the historical path of the packet, hence the reinforcement learning can be applied in the network routing.

According to the statements of Sutton [6], in the reinforcement learning, based on the five components of MDP, there are some parameters that needs to be defined. Firstly, the agent has two value functions, namely state-value function vπ(s) and action-state-value function qπ(s,a), which respectively describe the total expected reward of the agent that can be obtained from the environment from current state to terminal state under the π policy, and the total expected reward of the agent from current state to terminal state after a certain action is making under the π policy. Secondly, the total expected reward of these two value functions is defined as the expectation of the long-term reward, and the long-term reward is the weighted accumulation of all rewards from the current state to the terminal state. This implies that the farther the future state is from the current state, the smaller its impact on the long-term reward of the current state [6]. Hence, these two value functions can be written as

(1)

(2)

Where γ is discount rate.

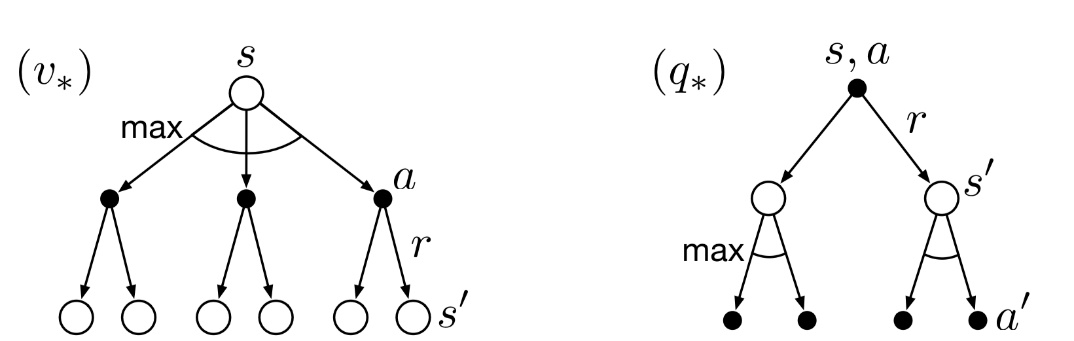
The value function is the objective function of the policy updating of the reinforcement learning, that is, the goal of the policy updating is to maximize (or minimize) the value function. The method of policy updating used in reinforcement learning is iterative method, which requires the use of the Bellman equation [6]. The Bellman equation is an important method for dynamic programming in the field of engineering and mathematics and it is introduced in reinforcement learning to reconstruct the value function [6].

Figure 2: The backup diagrams of vπ(s) and qπ(s,a) []

The figure 2 is the backup diagrams of vπ(s) and qπ(s,a), then the two value functions can be rewrote as follow

(3)

(4)

Where π(a|st) is the probability of action a at state st under policy π, is the reward of agent moving from s to s’, and is the probability of agent moving from s to s’ after making action a.

Then substituting equation (4) into equation (3) the vπ(s) can be rewrote as

(5)

substituting equation (3) into equation (4) the qπ(s,a) can be rewrote as

(6)

The equation (5) and (6) are the famous Bellman functions, which implies that the vπ(s) and qπ(s,a) can be treated as two system of linear equations, then the value of this two value functions can be derived by iterative method [7]. When the vπ(s) or qπ(s,a) is known, then the policy can be optimized by updating the action

, , (7)

With the new policy π(s), the new vπ(s) and qπ(s,a) can also be derived, then the policy can be optimized again. It can be proved that after finite iterations, the policy will converge to be optimal [7]. This way of policy updating is called policy iteration, which is a commonly used reinforcement learning method and because reward and state transfer probability are known this kind of learning process is called model based reinforcement learning.

**Model free reinforcement learning model**

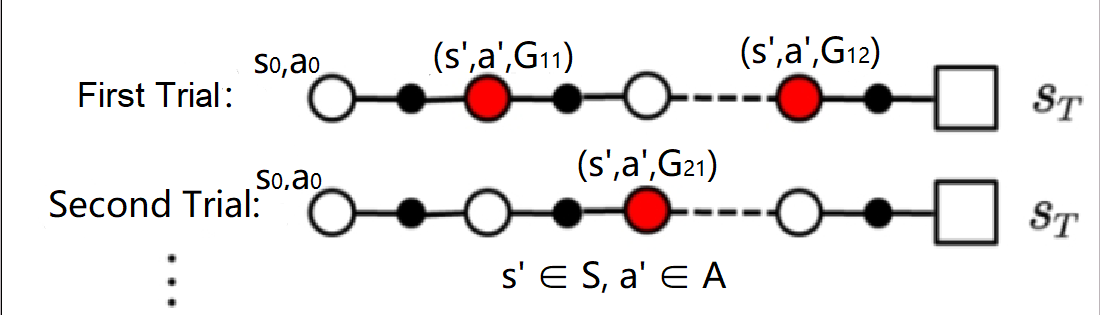
In a reinforcement learning model, when the rewards and state transition probabilities are unknown, the learning process of this model is called model free reinforcement learning. Model free reinforcement learning cannot use the iterative method that mentioned in the previous section to derive the value of the state value functions, but there are anther ways to evaluate the value functions. According to the definition of the state value functions, the right sides of these two equations are expectations, then the Monte Carlo method can be used to approximate the value functions.

Figure 3: Trial sequences of a reinforcement learning model

The idea of Monte Carlo method is to trial the model to obtain test data, and then use the test data to estimate real value. In the reinforcement learning problem, the agent will be initialized several times on different state-action pairs, then the agent will make actions under the current policy until the it reaches the terminal state. Finally, this method will get a set of trial sequences, as shown in Figure 3. After obtaining the trial sequence, calculating the value function of all state-action pairs that appear in the sequence

𝑎’∈𝐴, s’∈𝑆 (8)

When the value functions of all state-action pairs are estimated, the policy can be updated

𝑎∈𝐴, s∈𝑆 (9)

In the Monte Carlo method, the updated policy is the ε-greedy policy, that is, the action with the smallest value function of state s has the largest probability of being selected, and the other actions have equal smaller selected probabilities. The purpose of this is to ensure the model to be exploratory, otherwise, there will be some state-action pairs that may never be selected [8]. This is an important difference between the model based type reinforcement learning and the model free type reinforcement learning.

The Monte Carlo method is not a commonly used reinforcement learning method because it always needs the complete trial sequences to estimate the value function, which is not efficient. There is a more efficient and more used method called Q learning. The derivation of it is shown as following

The estimation equation of value functions in Monte Carlo method can be rewrote as

Then replacing 1/k to a constant parameter α, which means that the estimation becomes a moving average

(10)

(11)

Then the s’ and a’ can be replaced by st and at

(12)

Where is the new value of . Next splitting the to two parts

(13)

Equation (13) is the core idea of Q learning, that is, using the maximum value of the value function of the next state to estimate the value function of the current state [9]. In the Q learning, the trial and estimation of value functions can be performed simultaneously, which is more efficient than the Monte Carlo method [9].

**Reinforcement learning in network routing**

The reinforcement learning model can be easily established in a network routing problem. The network topology can be treated as the environment, each router in the network can be treated as each state, each packet forwarded in the network can be treated as an individual agent, the choice of next-hop of forwarding can be treated as the action, and the routing table of each router can be treated as the policy. For different types of packet, there are different types of rewards and value functions. For UDP packet, the reward can be delay between adjacent routers and the value function can be the end-to-end delay. For TCP packet, the reward can be link reliability between adjacent routers and the value function can be the end-to end packet loss rate. There are many proposed reinforcement learning routing method, the Q routing based on the Q learning is the most basic and most used method.

**Proposed Work and Approach or Methodology**

The Q routing model was proposed by Boyan and Littman [10] in 1994, which is based on the Q learning model.

(14)

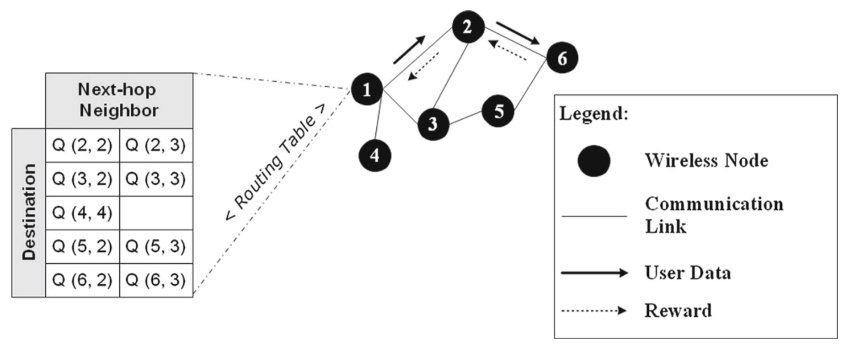
Equation (14) is the key idea of Q routing, where i is the router serial number where the packet is currently located, j is the serial number of the neighbour router of ith router, and are the destination of the packet, and are the sets of the neighbour routers of ith and jth router. The Q values of all destination-neighbour pairs of each router are stored in their routing tables, see figure 4.

Figure 4: The Q routing model [3]

For example, when a packet arrives node 1, it checks the destination of this packet, which is 6, then it searches its routing table and finds that the minimum Q value of destination 6 is Q(6,2). Next, node 1 forwards the packet to node 2 and when node 2 receives the packet it returns the delay between node 1 and node 2, which includes the queuing delay and the transmission delay, and it also returns its minimum Q value of destination 6. When node 1 receives the returned reward, it updates the Q value of destination 6.

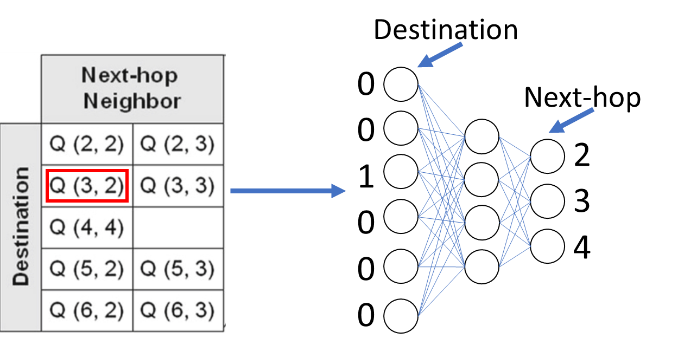
The Q routing has a drawback, that is, when the network scale is relatively large, the dimension of the Q routing table will become relatively large, which could affect the efficiency of searching and updating process and reduce the learning efficiency. Introducing deep Q network (DQN) into Q routing is a good solution. The DQN was proposed by DeepMind [11] in 2015, which is an enhanced reinforcement Q learning model combined with the deep learning. The biggest difference between the DQN and traditional Q learning is that the DQN uses the deep learning network to evaluate the value function, as shown in Figure 5.

Figure 5: The node 1 Q routing table and node 1 Q routing network [3]

For example, if node 1 receives a packet with destination 3, it will input 3 into its network in the form of one-hot and then the network will output the Q value of each neighbour of it. When node 1 receives the reward from one of its neighbour nodes, it will update its network by using gradient descent method or other deep learning updating methods. The deep learning network can improve the efficiency of searching and updating process, hence it is a popular research direction of Q routing. The DQN also has some other features, such as replay buffer, target network, double Q learning, and priority replay, which can also greatly enhance the performance of Q routing [11].

The replay buffer is a kind of buffer that stores rewards. When a node receives a returned reward from its neighbour node, the node does not directly use it to update the network but saves the reward into the replay buffer. When the buffer accumulates to a certain size, the node then randomly selects a batch of rewards in the buffer to update the network. The reason for this is that there is correlation between the samples in the trial sequences. For the deep learning model, if the training samples are not independently and identically distributed, the effect of the model training will be greatly reduced. Randomly selecting samples in the replay buffer can improve the independence of the training samples [11].

The target network refers to a network that has identical structure of the training network, but the update of the weights of this network is lagging behind the training network. In the Q routing, the update of the destination-neighbour value function of a node needs to use the destination-neighbour value function of its neighbour node. If the value function of its neighbour node changes too frequently, the update of the value function of the node will be difficult to converge. The aim of the target network is to solve the problem that the value function of neighbour nodes changes too frequently. At the start of the training, the target network duplicates the structure and weights of the training network, then then target network is frozen for a fixed period of time. After the frozen period the target network again duplicates the weights of the training network and then it is frozen again. During the frozen period, the target network is responsible for returning the value functions, so that the destination-neighbour value function of the neighbour nodes will not change too frequently, which is beneficial to the convergence of the node network [11].

Double Q learning refers to the separation of the action evaluation and the action selection in the Q learning. The long-term reward of a state-action pair of a state depends on the maximum value function of the next state, see equation (13). The max operation may make an over estimation to the value function and eventually the converged policy may not be the optimal policy. Hasselt [12] proposed a method called double Q learning which are intended to solve this problem. Double Q learning separates the action evaluation and action selection by using different value functions. In the Q learning, the Q target is

(15)

The action selection means that selecting an action a’ that has the largest state-action value, and the action evaluation means that using the state st and the selected action a’ to construct Q target. It can be seen that the selection and evaluation of the action are under the same policy π. In the double Q learning the selection and evaluation of the action use different value functions

(16)

Equation (16) is the key idea of double Q learning, the action selection is under the policy π and the action evaluation is under the policy π’. In the DQN, there are two networks which are the training network and the target network, so the node can use the training network for the action selection and use the target network for the action evaluation.

Priority replay refers to the use of weighted selection instead of uniform selection when randomly selecting training samples in the replay buffer. Schaul [13] first proposed the principle of the priority replay. Before the node puts a returned reward into the replay buffer, a weight of the reward is calculated and placed in the replay buffer along with the reward. During the selection of the training samples, the reward with a greater weight has a greater probability of being selected. The principle of the priority replay is shown as below.

The weight of the reward is defined as the absolute value of the Q error

(17)

And the probability of being selected is defined as

(18)

Where α is a constant parameter and 0< α <1, and k is the size of the batch of samples.

When the node uses the probability distribution of the priority replay, the evaluation of the action-state value function is biased, so the selected sample needs to multiply an importance sampling factor [13]

(19)

Where β is a constant parameter and 0< β <1.

The priority replay can make the rewards with larger errors have larger probabilities to be learnt, which can improve the learning efficiency of the model.

**References**

1. K. Wakefield, *A guide to machine learning algorithms and their applications Project* [Online]. Available: <https://www.sas.com/en_gb/insights/articles/analytics/machine-learning-algorithms.html> [Accessed on 19/06/2019]
2. E. O. Neftci and B. B. Averbeck, “Reinforcement learning in artificial and biological systems”, *Nature Machine Intelligence*, vol. 1, pp. 133-143, 2019, DOI: 10.1038/s42256-019-0025-4.
3. H. A. A. Al-Rawi et al., *“*Application of reinforcement learning to routing in distributed wireless networks: a review”, *Artif Intell Rev*, vol. 43, pp. 381-416, DOI: 10.1007/s10462-012-9383-6.
4. A. Foster, “Machine learning techniques applied to wireless ad-hoc networks: guide and survey”, *Proceedings of ISSNIP 3rd international conference intelligent sensors, Networks and information*, pp.365-370, 2007.
5. O. Alagoz et al., *“Markov Decision Processes: A Tool for Sequential Decision Making under Uncertainty”*, Medical Decision Making, 2010, DOI: 10.1177/0272989X09353194.
6. R. S. Sutton and A. G. Barto, Reinforcement Learning: An introduction, The MIT Press, 2nd edition, pp. 47-60, 2018.
7. R. S. Sutton and A. G. Barto, Reinforcement Learning: An introduction, The MIT Press, 2nd edition, pp. 73-88, 2018.
8. R. S. Sutton and A. G. Barto, Reinforcement Learning: An introduction, The MIT Press, 2nd edition, pp. 91-115, 2018.
9. R. S. Sutton and A. G. Barto, Reinforcement Learning: An introduction, The MIT Press, 2nd edition, pp. 131-132, 2018.
10. J. Boyan and M. Littman, “Packet routing in dynamically changing networks: a reinforcement learning approach”, *Proceedings of NIPS Adv neural information processing systems*, pp. 671-678, 1994.
11. V. Mnih et al., “Human-level control through deep reinforcement learning”, *Nature*, vol. 518, pp. 529-533, 2015, DOI: 10.1038/nature14236.
12. H. V. Hasselt et al. (8 Dec 2015), *Deep Reinforcement Learning with Double Q-learning* [Online]. Available: <https://arxiv.org/pdf/1509.06461.pdf> [Accessed on 19/06/2019]
13. T. Schaul et al., “Prioritized Experience Replay”, *ICLR 2016*, 2016