Self-selection Protocol Algorithm for Wireless Networks based on DDQN

Chongyang Wang¹, Hairong Yan¹* and Fan Li²

¹ Department of Information, Beijing University of Technology, Beijing, China
¹* Department of Information, Beijing University of Technology, Beijing, China
² Institute of economics and management, Beijing University of Technology, Beijing, China

*Corresponding author’s e-mail: yanhairong@bjut.edu.cn

Abstract. Given the current complex heterogeneous network environment, choosing different networks in different user scenarios limits the effectiveness of network access. To solve this problem, an algorithm based on DDQN is proposed to optimize the network configuration. The reward value is determined by service type, throughput, delay, and signal strength. DDQN can avoid the ping-pong effect that can be generated by network selection and stabilize the network environment by splitting action selection and action evaluation to prevent overestimation. The experimental results show that the algorithm can autonomously select and assign network protocols and can more accurately find network access methods suitable for user scenarios than the current network switching method represented by a single network parameter based on RSSI and CCA.

1. Introduction

With the development of network technology, there is a large amount of overlap between the widely used network technologies in the world today. The current network environment WLAN and cellular networks are the most common heterogeneous network combinations, which also play an important role in modern information and communication, and ISP also deploys their WLAN hotspots in dense user areas such as shopping malls, schools, and office buildings to disperse the pressure caused by cellular networks.

The next-generation heterogeneous network is a network that incorporates multiple protocols but has a complex environment that requires reliable network services to users at any time and any place. However, before this can be achieved, the network environment needs to be mature, and the functions of wireless network coverage, network self-configuration, and automatic management of network devices need to be solved. In the existing network environment, it is still difficult to achieve the above configuration with a single network protocol, but some algorithms can be used to achieve comprehensive resource scheduling for the current heterogeneous networks, and efficient use of heterogeneous network resources switching will gradually become a hot spot for research [1]. With the further development of wireless communications, there will be certain requirements for scalability and flexibility of heterogeneous networks, in paper [2], the authors propose a method of reinforcement learning algorithms on the provisioning of heterogeneous network resources, which improves the network resource provisioning capability and maximizes the network resource utilization.
In traditional heterogeneous networks, data transmission is often deeply tied to the control layer, which determines the mode of transmission, making it very complex to support flexible control policies. The SDN (Software-defined networking) controller performs data load balancing to improve the performance of the network. The new SDN-based wireless network architecture allows the entire network to be managed by a controller, which can be programmed to provide flexible control functions, including access control functions [3].

Reinforcement learning, as a tool that can make decisions to meet the requirements of the development environment in a non-deterministic environment, can make targeted adjustments according to the dynamic changes in the network, enabling heterogeneous wireless networks to become a solution that automatically adapts to changes in user scenarios and optimizes the network environment. Reinforcement learning is a type of machine learning in which an agent is constantly adjusted in an environment to maximize a specific metric. In wireless networks, due to the movement of nodes and the mutual interference between nodes, the network environment becomes complicated. Compared with traditional machine learning algorithms, the reinforcement learning algorithm’s learning speed and decision-making performance are effectively improved by pre-processing the data, directly extract features, and use the obtained historical data as training data [4].

In this paper, a DDQN-based network protocol switching algorithm is designed to achieve autonomous judgment of the current network environment under unknown circumstances and adopt the most appropriate network protocol for network access, so that the heterogeneous network can be in a stable and optimal network environment.

2. Self-selection algorithm of network protocol based on DDQN

Please follow these instructions as carefully as possible so all articles within a conference have the same style as the title page. This paragraph follows a section title so it should not be indented [2-3].

2.1. DQN and DDQN

The difference between DQN and Q-learning is Agent, which in Q-Learning is a Q-table and in DQN is a deep neural network. The input to the neural network is the state or the result of observation, and the output value is the action that the agent can perform. To train the neural network, the objective function is the Q value corresponding to each action and the input value is the state in which each intelligence is present. Paper [5] specifies the advantages of DDQN over traditional DQN networks.

The DDQN uses two identical neural networks, one of which uses experience replay to update the network in the same way as the DQN; the other neural network is derived from a copy of the previous neural network after the last episode has been executed. The actual source of the Q values is the second neural network. In DQN, the Q value is updated from the next state of max(Q-value), which is shown in equation (1):

\[ R_t^{DQN} = R_{t+1} + \gamma \max_a Q(S_{t+1}, a; \theta^-) \]

This creates the problem that if for a given state, a large value is computed for each Q calculation, then the value that passes through the neural network each time will get higher and higher until there is a large difference between each output value. Then, if the value of state \( a \) is higher than the value of state \( b \), state \( a \) will be selected every time for state \( s \). So now consider if for some experience \( b \) is optimal for some \( s \), but in the previous training for state \( s \) the neural network has been trained to give an extremely high value for action \( a \) when given state \( s \). This makes it difficult to train the action corresponding to state \( s \) under certain conditions as action \( b \). So, the solution is to reduce the difference between the network output values.

By using a second neural network and copying the last episode from the first neural network, it is clear that the value obtained in the second neural network is smaller than that of the main network so that the Q value can be obtained from the second network:

\[ R_t^{DoubleDQN} = R_{t+1} + \gamma Q(S_{t+1}, \text{argmax}_a Q(S_{t+1}, a; \theta^-), \theta^-) \]
2.2. Model Design

To use reinforcement learning, three variables, state; action; and reward value, need to be identified for definition.

Definition of state space: The state space $S$ of a terminal at moment $t$ is defined as $s_{mn} \in S$. The state space is:

$$S = s_1, s_2, ..., s_{mn}$$  \hspace{1cm} (3)

Definition of state: The description of network metrics in heterogeneous networks usually uses throughput, delay, packet loss rate, network load, etc. to describe the network service state, and uses network signal strength, node distance, node power consumption, cost, signal-to-noise ratio to describe user characteristics. In this paper, we will use the average throughput $T$, delay $D$ signal strength $P$, and node distance $W$ to describe the network state, the network quality $\Phi$ can be expressed as:

$$\Phi = T \times D \times P \times W$$  \hspace{1cm} (4)

Definition of action: It is necessary to set the action space for the intelligent body to choose, and the action space is defined as:

$$A = a_1, a_2, ..., a_n$$  \hspace{1cm} (5)

Where $a_n$ denotes the n network protocol used by a node.

Definition of reward:

Each node has its specific service characteristics when it is created and will have its service type, and even in the same network environment, it will have different reward values correspondingly. Combined with the actual requirements, the node service types are divided into the following categories:

- High real-time requirements, time delay to be as low as possible, need to be transmitted at a high rate if the delay will be too large to affect the service implementation. Also, need a certain amount of throughput to ensure the reliability of data.
- Extremely high throughput requirements, not as demanding relative to Service 1 real-time requirements, requiring larger data traffic.
- High requirements for latency, need to cope with the network traffic in unexpected situations, minimize the latency and improve the user experience.
- Just enough throughput to ensure.

In the above four cases, there are differences in network parameter requirements for different service types, and these differences will later have an impact on the division of reward value weights. Considering the whole network as a whole, the ultimate goal will be to optimize the overall network quality by selecting the nodes to use the protocol, and the reward value is a function with a strong correlation with the network.

$$V_t = v_1, v_2, ..., v_n$$  \hspace{1cm} (6)

$V_t$ represents the state information of the network at time $t$, which is a subset of the network state space $\Phi$. Thus, for a particular service $B$, the network state $V_t$, the reward functions $R$ is expressed as:

$$R = f_B(V_t)$$  \hspace{1cm} (7)

Node access affects changes in network parameters, and when an action is performed, the network state needs to be measured and the corresponding reward fed back. An action is considered valid when the executed action leads to higher network throughput, lower latency, and stronger signal strength; conversely, an action is considered invalid when the executed action leads to lower network throughput, lower latency, and lower signal strength. Therefore, the average throughput $\alpha_{avg}$, average time delay $\beta_{avg}$, signal strength $\gamma$ is considered when calculating the reward.

The units and values of different network parameters usually differ significantly and need to be normalized by performing a linear transformation of all values to map the values between $[0,1]$. Use $\min - \max$ standardization to eliminate the effects of unit differences in data:

$$f_t = \frac{x_t - \min(x)}{\max(x) - \min(x)}$$  \hspace{1cm} (8)

The normalized network average throughput $f_t(\alpha)_{avg}$, average time delay $f_t(\beta)_{avg}$, signal strength $f_t(\gamma)$ is obtained separately by normalizing using the above equation.
Combining the above equations gives the reward function $R$:

$$R = \omega_1 f_1(\alpha)_{avg} + \omega_2 f_2(\beta)_{avg} + \omega_3 f_3(\gamma)$$

(9)

$\omega_1, \omega_2, \omega_3$ are the weights of the average network throughput, delay, and signal strength.

One of the biggest shortcomings of using DQN is that although the $\arg\max()$ method allows the Q-value to rapidly converge to the target, it is likely to lead to overestimation, and the so-called overestimation means that the algorithmic model we obtain has a large bias. To solve this problem, it is possible to eliminate the error by separating the target Q-value calculation and the target Q-value selection. The network information is in a discrete state and DDQN can handle the data in the discrete state very well.

In DQN two neural networks are used for implementation, Q-MainNet, and Q-target. Similarly, in DDQN two networks are used to operate, only the target Q-value is calculated differently. After data processing, the network data is obtained and processed into the format needed for the algorithm and given to the control layer for processing. Q-MainNet gets system status and reward values, Q-value update by equation (2), Store the generated results as a memory group $(s_t, a_t, r_t, s_{t+1})$ in the experience replay, When the capacity in the experience replay is large enough. Q-target uses probabilistic random sampling of $\varepsilon$ from the experience replay. Then the deviation of the Q values of the two networks is minimized by a gradient descent algorithm. the Q-MainNet network learns and updates the parameters in real-time. Afterward, the network parameters are copied to the Q-target network after every G-step.

![Diagram](image)

Figure 1. Algorithm running process.

2.3. Model Simulation and Result Analysis

In this paper, the network environment design through NS3, for the same access point, design a movable node linear movement, simulate the impact of different distances in the vicinity of the same access point on the network, accessible protocols for WLAN and LTE, computing on the access node, NS3 approach is different from the real physical layer, the sending and receiving of data through NS3 by calling the object pointer to deliver Packet to achieve. Resource collection for statistics by setting callback functions in the simulation program. The DDQN algorithm will be built in Python using gym, stable_baselines3. To simulate a heterogeneous network access point will be used as a controller for network access control, the network topology consists of a control node and 10 random mobile nodes to simulate the change of the network state in the case of a movable network and interference generated between the nodes, whose positions are randomly distributed. Control nodes are used to perform calculations and apply calculation
results, 10 control nodes can be effective enough to simulate the signal interference in the multi-node state, The simulation environment parameters as Table1:

| Parameter Name                  | Symbol | Value  |
|---------------------------------|--------|--------|
| Frequency Band                  | $f$    | 5.51GHz|
| Power                           | $P$    | 21dBm  |
| Loss factor                     | $\delta$ | 3      |
| RSS Threshold                   | $RSS$  | -90dbm |
| Noise                           | $L$    | -95dbm |
| DDQN Learning Rate              | $\alpha$ | 0.1    |
| DDQN Attenuation factor         | $\gamma$ | 0.9    |
| DDQN Network update frequency   | $G$    | 4      |

The size of the Discount Factor determines how much importance the algorithm places on the future impact. The smaller the Discount Factor the more the algorithm tends to have a high short-term gain, but for the stability of the network 0.9 is chosen here to consider the long-term gain. In equation (9), $\omega_1, \omega_2, \omega_3$ The values are set to the weights between $[0,1]$ according to the service type of each node according to Table1.

The network architecture is designed as shown in Figure2 and based on the above design, the control nodes are used for data processing analysis and operations, and the throughput gradually stabilizes after training. It is obvious from Figure3 that the average value of throughput is stable around $55Mbit/s$ under DDQN algorithm operation, while the throughput of control nodes without algorithm management in WLAN-related protocols only through channel adjustment has a large variation in throughput and cannot reach the maximum throughput.

3. Conclusion
To improve the stability of heterogeneous wireless networks, enhance the throughput of the network, and be able to maintain a more stable state in the constantly changing dynamic network, this paper proposes a Self-selection Protocol algorithm for Wireless Networks based on DDQN, by combining the excellent performance of reinforcement learning in the MDP process, the DDQN algorithm is selected to dissolve the problem of possible overestimation of deep reinforcement learning, and the network quality quantified as a function for the training of deep neural networks. The simulation results show that the wireless network with the DDQN algorithm can quickly stabilize the network and maximize the throughput of the network under the state of constant node movement, and gradually stabilize the optimal network configuration to effectively solve the wireless network access problem.
Acknowledgments
The authors wish to acknowledge Dr. Hairong Yan, for her valuable discussions and feedback. They would also like to thank the open-source network simulator provided by the ns-3 project for experimental support.

References
[1] Zhao Y, Zhou W, Zhu Q. Q-learning based heterogeneous network selection algorithm[M]//Recent Advances in Computer Science and Information Engineering. Springer, Berlin, Heidelberg, 2012: 471-477.
[2] Heterogeneous wireless access networks: Architectures and protocols[M]. Springer Science and Business Media, 2008.
[3] Xu F, Qiu C, Guo A, et al. Access control for software-defined heterogeneous wireless access network[C]/2016 16th International Symposium on Communications and Information Technologies (ISCIT). IEEE, 2016: 520-524.

[4] LI Zi-heng; MENG Chao. Resource Allocation Algorithm of Wireless Network based on Deep Reinforcement Learning [J]. Communications Technology, 2020, 53(08): 1913–1917.

[5] Van Hasselt H, Guez A, Silver D. Deep reinforcement learning with double q-learning[C]/Proceedings of the AAAI conference on artificial intelligence. 2016, 30(1).

[6] Wang S, Liu H, Gomes P H, et al. Deep reinforcement learning for dynamic multichannel access in wireless networks[J]. IEEE Transactions on Cognitive Communications and Networking, 2018, 4(2): 257-265.

[7] Wang J, Du C, Wang Y. LTE/WLAN Heterogeneous Wireless Network Access Control Research[C]/2017 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC). IEEE, 2017, 2: 238-241.

[8] Qianbin C, GUANG L, Ziyu L I, et al. Deep Reinforcement Learning-based Adaptive Wireless Resource Allocation Algorithm for Heterogeneous Cloud Wireless Access Network[J]. Journal of Electronics and Information Technology, 2020, 42(6): 1468-1477.

[9] Riley G F, Henderson T R. The ns-3 network simulator[M]/Modeling and tools for network simulation. Springer, Berlin, Heidelberg, 2010: 15-34.

[10] Liu Z, Elhanany I. RL-MAC: a reinforcement learning-based MAC protocol for wireless sensor networks[J]. International Journal of Sensor Networks, 2006, 1(3-4): 117-124.