MDQN: An Enhanced Deep Q-Network Approach for Wireless Resource Management in Industrial IoT

Long Yu, Na Yin*
Applied Technology College of Dalian Ocean University, Dalian 116300, China
E-mail: yulong1972@126.com, 15941162715@163.com
*Corresponding author

Keywords: Deep Q-network, wireless network, resource management, industrial internet of things, channel rate

Received: February 21, 2025

With the rapid growth of access devices, traditional resource management algorithms fall short in meeting practical needs. To address issues like poor coordination of wireless network resources and quality of service in conventional wireless resource management, this study starts from the actual Industrial Internet of Things (IIoT) network structure. Firstly, a more practical IIoT wireless network framework is proposed. Then, combining the Markov decision process, the depth Q - network algorithm is improved, and a novel wireless network resource management algorithm (MDQN) is put forward. The results show that after training, the performance of the proposed algorithm significantly improves, surpassing the Slotted Aloha Algorithm (SAA) and Randomized Algorithm (RA). It achieves high channel utilization and system sum - rate, and converges faster than the traditional depth Q - network algorithm. When the arrival rate exceeds 1.0, the channel utilization and average sum - rate of the proposed algorithm reach over 94% and 25 bits/s/Hz respectively, which are more than twice those of the SAA and RA. As the system bandwidth increases, the latency of all algorithms decreases, and the proposed algorithm has the lowest system waiting latency and computational cost among them. These results demonstrate the algorithm's ability to achieve dynamic wireless resource management in IIoT and promote efficient utilization of wireless network resources.

Povzetek: Predlagani algoritem MDQN omogoča učinkovitejše in hitrejše dinamično upravljanje brezžičnih virov v IIoT kot obstoječe metode.

1 Introduction

With the swift advancement of commercial 5G networks and wireless communication technology, the number of devices accessing the network has increased sharply. The Internet of Things (IoT) is driving the development of traditional industries and accelerating the consumption of network resources. As a core resource in wireless networks, the static allocation mode of spectrum resources is inadequate to fulfill the escalating communication requirements [1-3]. Meanwhile, the development of the information and communication industry has led to a surge in demand for energy and computing resources, and researchers around the world have also carried out research on this. In wireless communication networks such as the IoT, Industrial Internet of Things (IIoT), and wireless sensor networks, communication devices are widely distributed and numerous, and usually rely on battery power. The battery capacity is limited and needs to be replaced regularly, resulting in high costs [4-6].Therefore, many scholars are dedicated to studying the related fields of wireless network resource management (RSMG), in order to optimize resource allocation (RSAL), improve resource utilization efficiency, and thus address the challenges of resource consumption. The research efforts focused on wireless resource management, such as those optimizing resource allocation and improving resource utilization efficiency, are of paramount significance in driving the sustainable development of wireless communication technology and ushering in the era of comprehensive intelligent interconnection [7].

Several studies have advanced wireless network resource management: Naderalizadeh et al. utilized multi-agent DRL for distributed resource allocation, outperforming decentralized methods and rivaling centralized benchmarks in balancing user rates [8]. Pham Q V et al. applied the whale optimization algorithm for RSAL, achieving energy-efficient and secure power allocation [9]. Shen Y et al. leveraged graph neural networks for large-scale wireless RSMG, surpassing classical optimization via unsupervised learning [10]. Mohajer A et al. proposed a dynamic framework for 5G heterogeneous networks, optimizing energy efficiency through carrier and power allocation while maintaining coverage [11]. Yang H et al. developed an asynchronous federated learning system for drone networks, improving learning accuracy and execution speed [12]. Chen Y et al. introduced a DRL-based RSMG algorithm for IIoT, reducing long-term task delays [13]. Lu H et al. proposed a user behavior-driven virtual network RSMG method, enhancing vehicle communication quality and user experience [14].

To enhance performance in dynamic network loads and low latency scenarios, Chao J et al. introduced a real-time adaptive scheduling method based on Transformer. The results showed that this method outperformed existing methods across multiple key performance indicators, with statistically significant performance improvements, providing valuable insights for future applications in IoT and remote urban networks [15]. To mitigate the mutual interference between perception and communication caused by spectrum and hardware resource sharing, Qi Q et al. developed a deep learning-based solution to improve the overall performance of 6G wireless networks. Both theoretical analysis and simulation results confirmed the effectiveness and robustness of the proposed solution in 6G wireless networks [16].

In the variants of the DQN algorithm, Double DQN reduces Q-value overestimation by decoupling action selection from evaluation in the target Q-value calculation. Dueling DQN, by modeling the state value function and the advantage function separately, can more efficiently estimate Q-values [17]. Additionally, other reinforcement learning methods, such as the policy gradient algorithm (e.g., PPO) and the actor-critic algorithm (e.g., A3C), have shown potential in wireless resource management, enabling them to handle continuous action spaces and adapt to more complex resource allocation scenarios [18].The relevant work summary is shown in Table 1.

In summary, despite some progress in wireless network resource management, current research has notable limitations. For instance, multi-agent deep reinforcement learning fails to take into account the real-world IIoT architecture, the whale optimization algorithm remains unintegrated with the Markov decision process, and graph neural networks are unable to effectively address the dynamic characteristics inherent in IIoT systems.

The research has three main goals: first, to develop a new wireless resource management algorithm (MDQN) for dynamic IIoT management by integrating the Markov decision process into the deep Q-network, based on the actual IIoT network structure; second, to experimentally prove MDQN's superiority over existing algorithms in improving channel utilization, system performance, rate, and reducing latency; third, to analyze MDQN's performance under diverse network conditions, offering theoretical and practical support for its IIoT application.

Table 1: Summary of relevant work.

Author	Key research content	Key findings	Compared with MDQN algorithm
Naderializadeh N et al. [8]	Multi-agent deep reinforcement learning is used for distributed resource management and interference mitigation in wireless networks	Compared to the decentralized baseline, it has an advantage in the trade-off between average and 5th percentile user rates, and its performance is close to or even better than the centralized information theory baseline	This method is based on multi-agent deep reinforcement learning for distributed management, and there are differences in the algorithm design ideas and application scenarios of the two methods
Pham Q V et al [9]	This paper studies the whale optimization algorithm and its application in wireless network resource allocation	Power allocation that can achieve the tradeoff between energy efficiency and spectral efficiency as well as power allocation that maximizes safe throughput	The optimization objectives and algorithm principles are different
Shen Y et al [10]	Application of graph neural network to solve large-scale wireless resource management problems	Training with unmarked samples without supervision can match or even exceed classical optimization algorithms without domain-specific knowledge	The two are different in network structure and problem-solving methods
Mohajer A et al [11]	A dynamic optimization model is proposed to minimize the overall energy consumption of the fifth-generation heterogeneous network and provide the necessary coverage and capacity	While ensuring the throughput requirements of uniform and hot spot user equipment distribution mode, the power saving rate in different traffic models is considerable	This model is aimed at energy consumption optimization of fifth-generation heterogeneous networks, with different optimization objectives and applicable scenarios
Yang H et al [12]	Develop a framework for asynchronous federated learning for multi-drone networks	Higher learning accuracy and faster federated execution time are achieved	The application scenarios and algorithm advantages are different
Chen Y et al [13]	In view of the dynamic resource management problem of joint power control and computing resource allocation in ⅱOT, a dynamic resource management algorithm based on deep reinforcement learning is proposed	Can effectively reduce the long-term average delay of tasks	MDQN is also aimed at IIoT, but the algorithm design combines Markov decision process and improved deep Q network, which is different in optimization objectives and algorithm details
Lu H et al [14]	A virtual network resource management method based on user behavior is proposed to optimize vehicle communication	Can significantly improve service quality and experience	MDQN focuses on IIoT wireless resource management, which is tailored to different vehicle communications, application scenarios and optimization directions
Chao J et al [15]	A real-time adaptive scheduling method based on Transformer is proposed	It is superior to the existing methods in several key performance indicators	The algorithm structure and application scenarios are different
Qi Q et al. [16]	Develop deep learning-based solutions to improve the overall performance of 6G wireless networks	It is effective and robust in 6G wireless network	MDQN is a solution for wireless resource management in IIOT

This study targets designing an efficient wireless resource management algorithm for the IIoT's complex wireless environment, aiming to improve channel utilization, system performance, rate, and reduce latency. It proposes integrating the actual IIoT network structure, Markov decision processes, and the enhanced MDQN algorithm to optimize wireless resource management in dynamic networks and achieve performance goals.

In summary, although significant achievements are made in the field of wireless network RSMG, the current wireless networks are complex and random, making it difficult to predict and effectively manage. In the face of this challenge, the research needs to continuously explore and innovate more advanced RSMG strategies and methods. Based on this, the research innovatively starts from the actual IIoT network structure, combines Markov decision process, improves the Deep Q Network (DQN) algorithm, and proposes a new wireless network RSMG algorithm, MDQN, in order to successfully achieve dynamic wireless RSMG of IIoT and promote efficient utilization of wireless network resources.

2 Methods and materials

2.1 IIoT wireless network RSMG framework design

User Equipments (UEs) in IIoT are usually connected to each other through wireless communication, but wireless spectrum resources are limited, and traditional management techniques are difficult to achieve expected performance [19-20].To address this issue, research is conducted on optimizing wireless RSMG algorithms based on deep Q-networks, and an MDQN algorithm is proposed. Firstly, a more practical IIoT wireless network framework has been proposed, as shown in Figure 1.

Figure 1 illustrates that the IIoT model accommodates diverse UE types (e.g., sensors, industrial devices) with varying QoS and storage requirements, necessitating tailored system design. Data prioritization is critical, with emergency information (e.g., equipment failures, safety alerts) receiving higher processing priority for system stability. Additionally, many IIoT UEs operate cyclically rather than continuously, such as environmental or security monitoring devices, which transmit data at predefined intervals for analysis [references implied by original context] [21-23]. This periodic data transmission mode poses new challenges for RSMG and system optimization. For the first type of UE, if it has successfully accessed the communication channel and has not experienced any form of signal collision with other UE during transmission, the specific numerical description of the Signal to Noise Ratio (SNR) when the UE communicates is shown in formula (1).

$$SNR_k(i) = \frac{\sum_{n=1}^{N} P^f U_{kn}(i)\left|h_{kn}(i)\right|^2}{\sigma^2}, \quad k \in K_1 \qquad (1)$$

In equation (1), σ^2 represents the received noise power, U represents the binary identifier of the sub-channel. P^f represents the fixed transmission power of the i UE. Here, i is only used to identify different user equipment and has nothing to do with the channel. h represents small-scale fading, and N represents the set of channels. For the second type of UE, its probability P_k 's description formula is shown in formula (2).

$$\mathrm{P_k(l(i))} = \frac{\exp(-\gamma_k)(\gamma_k)^{\mathrm{l(i)}}}{\mathrm{l(i)!}}, \quad \mathrm{k} \in \mathrm{K}_2 \qquad (2)$$

In equation (2), $\mathrm{P_k(l(i))}$ represents the probability that the data count of the k second-class UE in set K_2 is $\mathrm{l(i)}$. The parameter γ_k indicates the distribution characteristics of the data count for the $\mathrm{k} =$ second-class user equipment, reflecting its data distribution. The data length of the memory at subsequent times is determined by equation (3).

$$d_k(i+1) = \min\left\{d_k(i) + l(i) - g_k(i), F\right\}, \quad k \in K_2 \qquad (3)$$

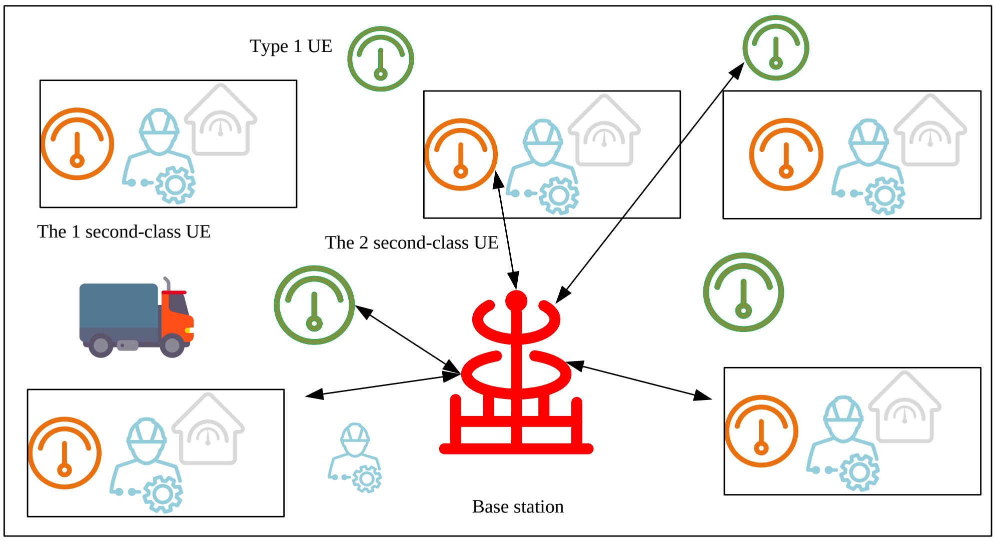


Figure 1: IIoT wireless network RSMG framework diagram.

In equation (3), $d_k(i+1)$ represents the memory data length of the k second-class user equipment in set K_2 at time $i+1$. $g_k(i)$ denotes the data length transmitted by the G-th second-class user equipment during the slot at time i, $d_k(i)$ is the memory data length of this device at time i, and F stands for the maximum capacity of the memory. Equation (3) primarily addresses the data length variations of the second type of user equipment. The study can be further expanded to include the first type of user equipment, where the memory data length changes can be similarly represented as $d_m(i+1)=\min\{d_m(i)+l_m(i)-g_m(i),F_m\}$, with each parameter explained in a similar manner. In future research, to simplify the model and highlight key points, the analysis will initially focus on the data length variations of the second type of user equipment, although the overall approach can be extended to resource management scenarios involving both types of user equipment. The optimization core of this study aims to further improve the spectrum utilization efficiency of IIoT systems while ensuring that the QoS requirements of two types of UEs are met. Specifically, the goal is to maximize the number of UEs that successfully establish communication connections, in order to achieve efficient utilization of spectrum resources. Meanwhile, for emergency data, the system needs to have the ability to respond quickly to ensure timely transmission and processing of data. Therefore, the wireless RSMG problem is shown in formula (4).

$$\begin{cases} \left\{\sum_k \left(O_k(i)+O_k^E(i)\right)\right\} \\ U_{kn}(i)\in\{0,1\},(k\in K,n\in N) \\ \sum_{n=1}^{N} U_{kn}(i)\le 1,(k\in K,n\in N) \\ \sum_{k=1}^{K} U_{kn}(i)\le 1,(k\in K,n\in N) \\ O_k^E=1,\left(k\in K_1\right) \\ W\log\left(1+SNR_k(i)\right)\ge V_0,\left(k\in K_1\right) \\ d_k(i)<F,\left(k\in K_2\right) \end{cases} \tag{4}$$

In equation (4), W is the bandwidth of the sub-channel and O_k^E is the feedback information of emergency data. $U_n(i)$ indicates the status of i user devices on the n th sub-channel. When it is 1, it means that the Gaussian user device is using the n sub-channel; when it is 0, it means it is not in use. n represents the sub-channel number, and K represents the set of sub-channels. $O=1,\left(k\in K_1\right)$ indicates that when k belongs to the first category of user devices, a certain status identifier is 1; $d_1(i)<F,\left(k\in K_2\right)$ indicates that when k belongs to the second category of user devices, the data length of the i first category user device is less than the threshold F.

To achieve efficient spectrum RSMG, Base Stations (BSs) must comprehensively obtain various environmental information in IIoT systems. These information include but are not limited to the allocation details of sub-channels, priority sorting of data, actual length of data in memory, and specific type of UE, which together constitute an important basis for BS to make RSMG decisions. To facilitate the BS to obtain this information, a simple Media Access Control (MAC) frame structure was designed to achieve the acquisition of environmental state information. The MAC frame structure is shown in Figure 2.

The MAC frame structure design integrates information such as UE type, data priority, and data length with actual data. Among these, UE type information is used by the BS to identify various user devices, such as sensors and industrial smart devices. Data priority information distinguishes between emergency data (such as equipment failure reports, safety alerts, and power shortage notifications) and regular data, with emergency data receiving a higher processing priority. Data length information helps the BS understand the size of the data, enabling it to allocate resources efficiently. The data priority set identifies different data priorities, and the BS uses this information to make decisions on spectrum resource management for the next time slot.

2.2 Action and reward function design

In IIoT industrial IoT systems, BS plays the role of the only intelligent agent, which can capture and analyze real-time status information of the entire IIoT system based on a carefully designed MAC frame structure. However, it should be noted that BS can only obtain detailed information of UEs that have successfully established communication links with it. For UEs that fail to communicate successfully, their relevant information is replaced with specific characters, such as 'N/A'. This method of substitution is similar to data imputation, which simplifies the data processing process. It allows the BS to focus more on analyzing the information of UEs that communicate successfully, while also ensuring the simplicity and accuracy of data processing. However, this approach may affect the accuracy of determining the status of UEs that fail to communicate successfully [19]. After the basic framework for the IIoT wireless network RSMG has been constructed successfully, in an effort to equip the BS with self - learning and decision - making abilities so that it can steadily grasp and implement effective RSMG strategies, research efforts have been directed towards enhancing the reward function. The construction of this function aims to guide BS to continuously optimize its RSMG decisions through a reasonable reward and punishment mechanism, to adapt to the dynamically changing IIoT environment. The principle of interaction between BS and environment is shown in Figure 3.

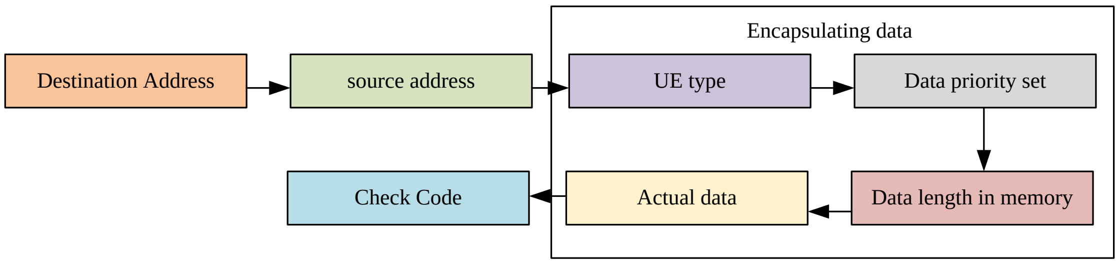


Figure 2: Schematic diagram of MAC frame structure.

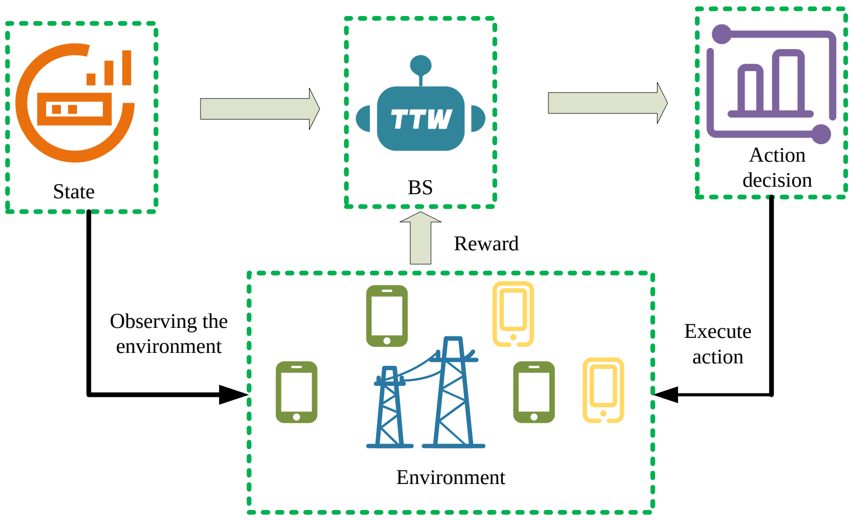


Figure 3: Schematic diagram of BS and environment interaction principle.

The optimization of wireless RSMG is refined into four sub-objectives to enhance system performance: 1) maximizing spectrum resource utilization to support more UEs and improve network communication capacity; 2) prioritizing rapid access and transmission of urgent data for timely emergency response; 3) ensuring stable communication by meeting rate thresholds for the first type of UE; and 4) guaranteeing timely data transmission for the second type of UE to prevent data overflow. Therefore, feedback information can be used to measure the quantum reward function $R_o(i)$, as shown in equation (5).

$$R_o(i) = \gamma_1 \sum_{k \in \mathrm{K}, k \acute{\mathrm{U}} \varsigma} O_k(i) + \gamma_2 \sum_{j \in \varsigma} O_j^E(i) \tag{5}$$

In equation (5), ς represents the set of UEs in an emergency situation, $|\varsigma|$ represents the total number of UEs in this set, and γ_1 and γ_2 represent weight coefficients used to balance the importance of different reward factors. To achieve resource optimization, the reward mechanism prioritizes emergency data and sets differences, while ensuring fair distribution. Research designs sub-reward functions to meet specific rate requirements, as shown in formula (6).

$$R_v(i) = \begin{cases} \sum_k V_k(i) \\ \sum_k O_k(i)^* \mu_1 \end{cases} \tag{6}$$

In equation (6), μ_1 represents the constant rate of the first type of user. The instantaneous rate here refers to the data transmission rate of the device at the current moment. Although the formula does not directly aim to maximize the number of successful communication connections established by UE, the design of the reward mechanism guides the BS to optimize resource allocation, thereby indirectly achieving this goal. $\sum_i \mathrm{O}_i$ represents the sum of the indicators indicating whether all first-class user equipment meet the rate threshold requirements. V_k represents the instantaneous rate obtained by UE. represents the instantaneous rate obtained by UE. The design of the sub-incentive function balances wireless RSMG optimization objectives with algorithmic learning efficiency. Although adopting a strategy that specifically targets the third sub-goal has the potential to accelerate its attainment, such an approach runs the risk of undermining the overall optimization process and the efficiency of the learning mechanism. The sub-reward remains zero until all first-type users meet bandwidth requirements, then jumps to a positive value. However, this approach may significantly delay DQN learning due to prolonged zero rewards during early training, as insufficient positive feedback hinders rapid algorithmic progress. Finally, to effectively avoid the problem of data overflow in the storage of the second type of UEs during data processing, a sub-reward function $R_d(i)$ was designed. The design of this sub-reward function is based on the clearly defined required memory length in the optimization problem. The specific function expression and design details are shown in equation (7).

$$R_d(i) = \begin{cases} \sum_k O_k(i)^* \mu_2 \\ \sum_k V_k(i), \end{cases} \qquad (7)$$

In equation (7), F_1 represents the predicted length in case of excessive memory data, which is obtained from historical data and current system load conditions. It is not a constant but a dynamic prediction value. μ_2 represents a value greater than the maximum rate of the second type of UE, which is used to measure the risk degree of data overflow. When $d_k(i) \succ F_1$, due to increased system load or surge in data traffic, the probability of data overflow in the memory will significantly increase, which means that the risk of data loss or processing delay will increase significantly. On the contrary, the sub-reward function is designed based on the sum rate of all second type UEs, aiming to optimize overall performance. Taking into account these factors, the reward function designed for the study is shown in equation (8).

$$R(i) = \gamma_o R_o(i) + \gamma_v R_v(i) + \gamma_d R_d(i) \qquad (8)$$

In equation (8) γ_o , γ_v , and γ_d respectively represent the weight factors for balancing the three sub-rewards.

2.3 MDQN algorithm design

After completing the design of the incentive function, to more validly solve the complex problem of wireless RSMG, the action space was carefully compressed. This step aims to decrease the complexity of computations involved in the algorithm while ensuring the rational allocation of wireless resources. In addition, the DQN algorithm was optimized using a prioritized experience replay approach utilizing Temporal Difference (TD) error. Through this series of improvements, a spectrum RSMG algorithm for MDQN was ultimately proposed. This algorithm aims to further improve the management efficiency of wireless resources to meet the growing communication demands. Traditional Q-learning can ultimately converge to the optimal Q-table through sufficient exploration, as shown in the learning flowchart in Figure 4.

This Figure 4 shows the basic principle and process of DQN algorithm. MDQN algorithm is an improvement on DQN algorithm, which is based on action space compression and priority experience replay strategy based on TD error. The traditional DQN algorithm converges slowly and requires a large amount of storage space when the state space and action dimensions are large. In response to this, the study first compresses the action space to reduce complexity and avoid collisions, while fully allocating channels. Action space compression technology reduces algorithmic computational complexity while ensuring efficient wireless resource allocation in IIoT. The original action space, with numerous resource allocation combinations, increases computational load and slows convergence. This is achieved through: 1) Device priority and data feature screening, prioritizing devices with urgent information. 2) Channel state pre-assessment, retaining only sub-channels with good states. 3) Action aggregation, combining similar resource allocation actions. These methods reduce the action space's dimensionality and complexity. The dimension of the newly compressed action space is reduced. To improve learning efficiency, a priority experience replay strategy based on TD error is proposed, which measures the importance of historical experience data and prioritizes replay of highly important data to accelerate NN learning. The pseudocode of the implementation of the priority experience replay strategy based on TD error is shown in Figure 5.

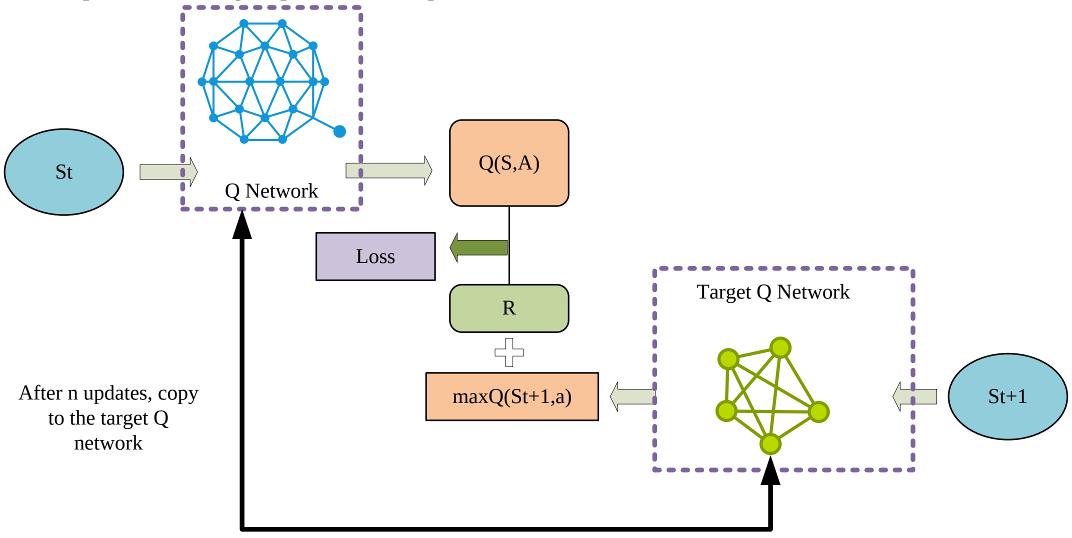


Figure 4: Principle diagram of MDQN algorithm.

```
class PrioritizedReplayBuffer:
    def __init__(self, capacity, alpha, beta):
        self.capacity = capacity
        self.alpha = alpha  # prioritization strength
        self.beta = beta    # importance sampling compensation
        self.priorities = SumTree(capacity)
        self.memory = []

    def store(self, experience):
        max_priority = self.priorities.max() if self.memory else 1.0
        self.memory.append(experience)
        self.priorities.insert(max_priority)

    def sample(self, batch_size):
        indices = []
        priorities = []
        segment = self.priorities.total() / batch_size

        for i in range(batch_size):
            s = random.uniform(segment*i, segment*(i+1))
            idx, priority = self.priorities.get(s)
            indices.append(idx)
            priorities.append(priority)

        sampling_probs = priorities / self.priorities.total()
        weights = (len(self.memory) * sampling_probs) ** -self.beta
        weights /= weights.max()

        return indices, weights

    def update_priorities(self, indices, td_errors):
        for idx, td in zip(indices, td_errors):
            priority = (abs(td) + 1e-6) ** self.alpha
            self.priorities.update(idx, priority)
```

Figure 5:Pseudo-code diagram of the implementation of priority experience replay strategy based on TD error.

Figure 5 illustrates that the TD error-based prioritized experience replay strategy begins by calculating TD errors (differences between predicted and target values) for each training sample. Sample priorities, proportional to absolute TD errors, guide selection from the replay buffer, favoring higher-priority samples. Importance-sampling weights correct sampling bias and are applied during neural network updates to emphasize high-priority samples' influence.

The importance indicator of empirical data is determined based on TD error.These improvements aim to overcome the shortcomings of traditional DQN and enhance algorithm performance and resource utilization efficiency. The description of TD error is shown in equation (9).

$$\begin{aligned}\delta_i &= Q_{\text{target}}(\mathbf{s}(i),\mathbf{a}(i)) - Q(\mathbf{s}(i),\mathbf{a}(i)) \\ &= R(i) + \varphi \max_{\mathbf{a}} \hat{Q}\left(\mathbf{s}(i+1),\mathbf{a},W^-,b^-\right) - Q(\mathbf{s}(i),\mathbf{a}(i),W,b)\end{aligned} \quad (9)$$

In equation (9), $\mathbf{s}(i)$ represents the system environment state, $\mathbf{a}(i)$ represents the sampling action, b represents the network parameters, $\mathbf{a}(i)$ represents the action corresponding to the maximum Q value, that is, the wireless RSMG result. The larger the TD error of the verification data, the stronger its importance. When initially stored in the experience memory, the maximum importance value is assigned, and the TD error is reduced in the later stage. To prevent duplicate sampling, the TD error is redefined in the study, as shown in equation (10).

$$I_i^m = |\delta_i|\left(\tau_{\text{ini}} + \left(\tau_{\text{end}} - \tau_{\text{ini}}\right) * \exp(-\vartheta * i)\right) \quad (10)$$

In equation (10), τ_{ini} represents the initial discount factor. To guarantee the consistency and efficiency of the algorithm, parameter ϑ's setting is crucial. It should be kept on the same order of magnitude as the learning rate and delay rate, so that the TD error can change at a reasonable speed and optimize the learning process. In the MDQN algorithm, hyperparameter selection is carefully considered and experimentally validated. The learning rate is set to balance fast convergence and stability, with 0.0001 chosen after multiple experiments. The initial discount factor is selected to balance short-term and long-term rewards, crucial for IIoT wireless resource management. After experimental optimization, it is determined to ensure reasonable resource management decisions in dynamic networks. These selections result from experimental exploration and performance evaluation, aiming to balance algorithm performance and resource utilization efficiency. Finally, a wireless RSMG strategy based on MDQN was developed, as shown in Figure 6.

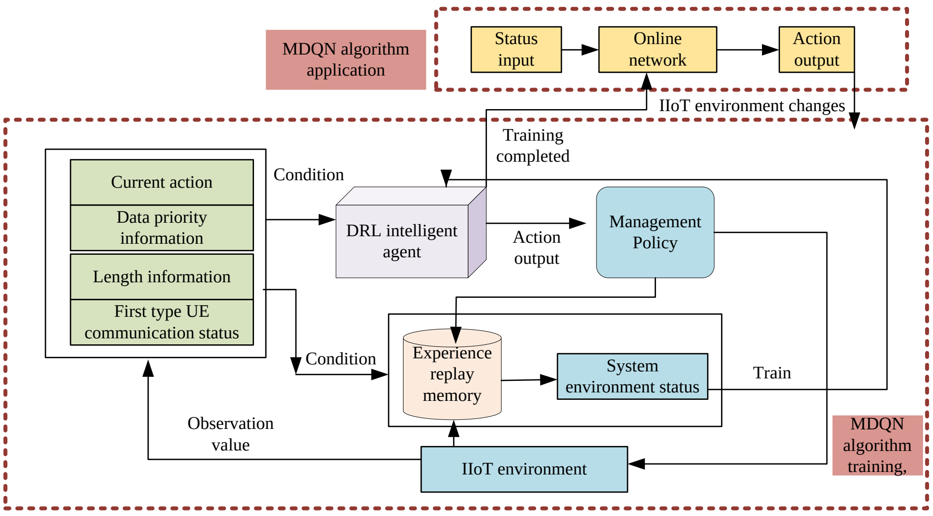


Figure 6: Schematic diagram of wireless RSMG strategy based on MDQN.

```
class MDQNAgent:
    def __init__(self, state_dim, action_dim):
        self.main_net = QNetwork(state_dim, action_dim)
        self.target_net = QNetwork(state_dim, action_dim)
        self.replay_buffer = PrioritizedReplayBuffer(capacity=100000)
        self.epsilon = 1.0
        self.gamma = 0.99
        self.tau = 0.005

    def remember(self, state, action, reward, next_state, done):
        # Store new experience with initial priority
        experience = (state, action, reward, next_state, done)
        self.replay_buffer.store(experience)

    def _calculate_td_error(self, experience):
        state, action, reward, next_state, done = experience
        current_q = self.main_net(state)[action]
        target_q = reward + (1 - done) * self.gamma * self.target_net(next_state).max()
        return target_q - current_q

    def _sample_batch(self, batch_size):
        # Returns indices, batch, importance sampling weights
        return self.replay_buffer.sample(batch_size)

    def train(self, batch_size):
        if len(self.replay_buffer) < batch_size:
            return

        indices, batch, weights = self._sample_batch(batch_size)
        td_errors = []

        for experience in batch:
            td_error = self._calculate_td_error(experience)
            td_errors.append(td_error)

        # Update main network with prioritized experience
        loss = self._compute_loss(batch, weights)
        self.optimizer.zero_grad()
        loss.backward()
        self.optimizer.step()

        # Soft update target network
        self._update_target_network()

        # Update experience priorities
        self.replay_buffer.update_priorities(indices, td_errors)

    def act(self, state):
        if random.random() < self.epsilon:
            return random.randint(0, self.action_dim - 1)
        return self.main_net(state).argmax().item()
```

Figure 7: Pseudo code of MDQN algorithm.

In Figure 6, during the training process, the neural network continuously adjusts its parameters to gradually reduce the loss function, ultimately converging to the optimal parameters. During application, BS monitors the data packet headers to obtain the environmental state and uses the trained neural network to predict the state of the next time slot. The deep neural network (DNN) selects the action corresponding to the highest Q value to manage spectrum resources. The results are broadcast to enable UE access to the channel, achieving resource management based on the optimal strategy. This convergence is achieved through extensive training data and iterative processes. The pseudocode of MDQN algorithm is shown in Figure 7.

Deep reinforcement learning algorithms like MDQN are often seen as "black boxes." To enhance interpretability, saliency maps can be introduced, which visualize the importance of input features (e.g., device or channel status in IIoT) to MDQN's decision-making by calculating gradients of input states on output actions. A high saliency value for a feature indicates its significant influence on resource allocation, helping to demystify MDQN's decisions and improve its credibility and practicality.

3 Results

3.1 Experimental environment and model training

Strengthening the performance verification of spectrum management algorithms plays an important role in analyzing spectrum utilization and resource block stability. Based on the IIoT wireless network RSMG system, experimental analysis was conducted using the Tensor Flow framework. After iterative training of DNN, the average performance was tested through multiple runs. This experiment selected three wireless RSMG algorithms based on DQN, Slotted Aloha Algorithm (SAA), and Randomized Algorithm (RA) as comparison algorithms. Channel utilization (CU), system, and rate were selected as evaluation indicators. Here, average data rate is considered as a key performance metric reflecting the efficiency of data transmission in the system, which is closely related to channel utilization but provides a more specific measure of the system's communication capability. The experimental operating environment is in Table 2. The study first conducted a hyperparameter sensitivity analysis, and the results are shown in Table 3.

As shown in Table 3, the sensitivity analysis of key hyperparameters in the MDQN algorithm, including the learning rate, discount factor, and batch size, revealed that increasing the learning rate from 0.001 to 0.01 enhanced channel utilization, system performance, and rate, while reducing system latency. The performance gradually improved as the discount factor increased from 0.8 to 0.95, and the algorithm's performance significantly improved when the batch size increased from 32 to 128. This indicated that the settings of these hyperparameters significantly impacted the performance of the MDQN algorithm, and appropriate optimization could enhance its wireless resource management capabilities.

The experimental results are the average values after a single training session. When considering the network function values, the initial state of IIoT will be randomly set in each validation test to ensure the comprehensiveness and accuracy of the validation. The training effect of the raised algorithm is in Figure 8.

Table 2: Experimental operating environment.

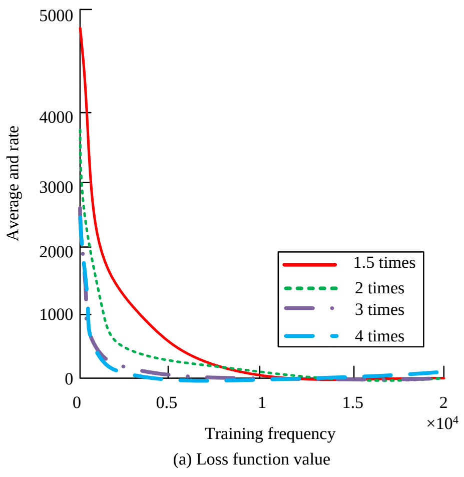
parameter	Experimental environment
Processor	11th GenIntel(R)Core(TM)i5-1135G7@2.40GHz-2.42GHz
Memory capacity	4GB RAM
Operating system	Windows7
Data mining software	SPSS Modeler18.0
Programming environment	Python3.8.3
Programming IDE	Anaconda3
Model building	Python3.8.3

Table 3: Hyperparameter sensitivity analysis.

Hyperparameter	Short-cut process	Channel utilization	System and rate (bits/s/Hz)	System delay (ms)
Learning Rate	0.001	0.88	22	0.35
	0.005	0.90	23	0.32
	0.01	0.92	24	0.30
Discount Factor	0.8	0.89	23	0.33
	0.9	0.91	24	0.31
	0.95	0.93	25	0.29
Batch Size	32	0.87	21	0.36
	64	0.90	23	0.32
	128	0.92	24	0.30

The results in Figure 8 indicated that an increase in the number of channel resources correspondingly resulted in a reduction of the loss value of the algorithm and a rise in the reward value. The results in Figure 8 (a) indicated that the loss value of the MDQN algorithm showed a stable curve trend in the later stage of training, and the larger the number of channel resources, the faster it tended to stabilize. In Figure 8 (b), the reward value initially increased rapidly and stabilized after 2500 training sessions, reflecting the learning efficiency and convergence of MDQN. The fewer sub-channels there were, the smaller the reward value, which led to a decrease in system performance due to a reduction in successful UE access. Some data points were missing due to experimental errors during data collection. These missing data points were taken into account in the subsequent analysis and did not significantly affect the overall results. The CU, system, and rate test results of the MDQN algorithm proposed by the research institute compared to DQN, SAA, and RA are shown in Figure 9.

From Figure 9, in the early stages of training, the performance of MDQN was similar to that of SAA, but it showed a significant improvement over SAA after a certain number of training iterations. As shown in the graphs, MDQN eventually outperformed SAA. With the deepening of training, MDQN algorithm showed its excellent performance advantages. It could achieve channel utilization close to 90%, and provided an average system and rate of about 25bits/s/Hz, which was three times that of SAA and four times that of RA. Furthermore, in comparison with the DQN algorithm, MDQN also demonstrated its outstanding performance. MDQN not only had faster convergence speed, but also performed better. The reason was that the improvement of error and excitation function in MDQN increased its CU and system application rate, greatly optimizing the calculation steps of action space. Ablation experiments assessed component contributions to the MDQN algorithm's performance. Three variants were tested: MDQN-NoER (without experience replay), MDQN-NoTDP (without TD-based priority), and full MDQN (with both). Evaluated in the same IIoT environment using channel utilization, system performance, rate, and delay metrics across 10 trials, results are in Table 4.



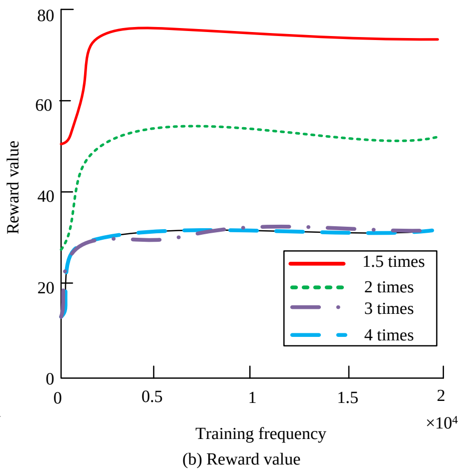
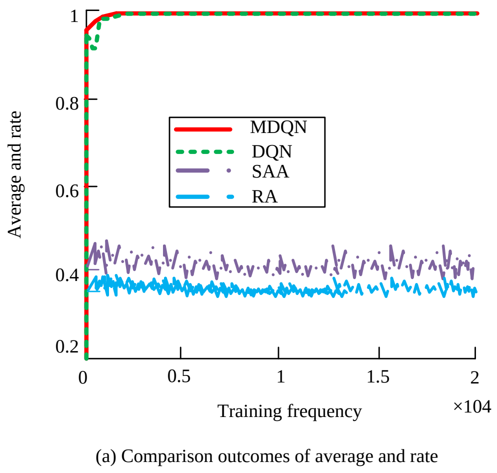


Figure 8: Training effect diagram of the research algorithm.



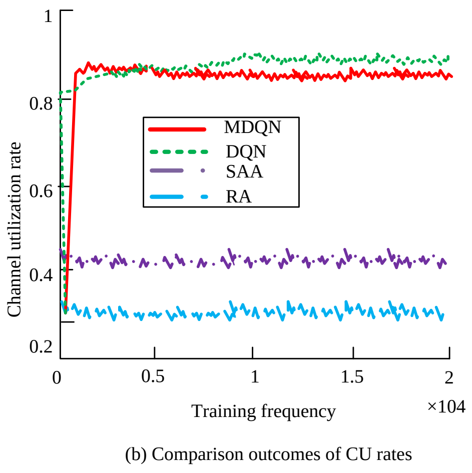


Figure 9: Comparison of training effects of different algorithms.

Table 4: Ablation results.

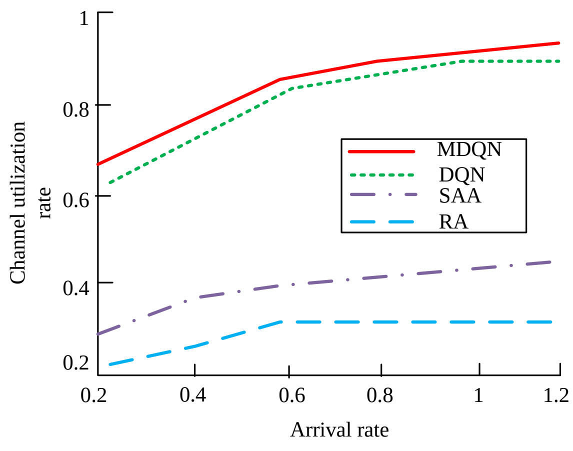
Algorithma	Channel utilization	System and rate (bits/s/Hz)	System delay (ms)
MDQN-NoER	82.1%	18.5	0.38
MDQN-NoTDP	86.7%	21.3	0.32
MDQN(Complete algorithm)	91.5%	23.2	0.27

Table 4 shows that removing experience replay (MDQN-NoER) severely harmed algorithm performance, reducing channel utilization, system efficiency, and rate while increasing latency, highlighting its critical role. Removing TD-based priority (MDQN-NoTDP) also degraded performance but less so, indicating its lesser impact. The full MDQN, combining both mechanisms, achieves optimal performance with the best metrics, demonstrating the importance of their synergy.

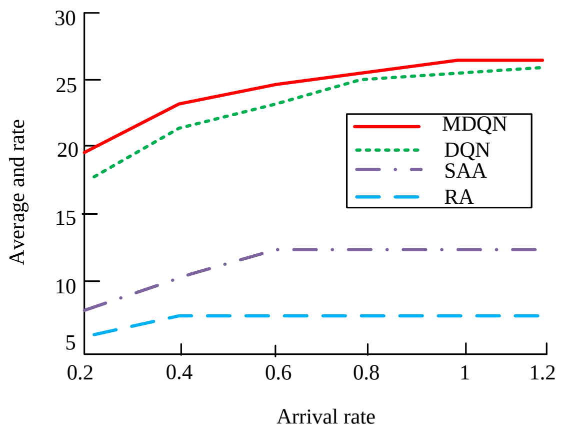
3.2 Model performance testing

During the training and application process of DQN, various problems such as overfitting, under fitting, and sample bias may occur. Research on improving the DQN algorithm and conducting robustness analysis can help identify these problems and improve the credibility of the algorithm. During the testing process, it is necessary to analyze the resource situation in its initial state. Factors such as channel conditions, data requirements, and number of devices can all affect the management of DQN resources. Channel conditions (e.g., gain and noise) directly affect signal transmission quality, with poor conditions increasing error rates and requiring DQN to adjust resource allocation for reliability. Data requirements varied by priority, demanding tailored latency and bandwidth, while growing device numbers intensify resource competition, compelling DQN to optimize allocation efficiency. Figure 10 shows the CU and system application of various algorithms.

Figure 10 (a) indicates the comparison results of CU rates of different algorithms at arrival rates, and Figure 10 (b) indicates the comparison results of average rates of different algorithms at arrival rates. From Figure 10 (a), in the test of channel utilization, a key metric, the channel utilization of each algorithm generally increased with the arrival rate. When the arrival rate was 0.4, the MDQN algorithm achieved a channel utilization of about 60%, the DQN algorithm about 40%, the SAA about 30%, and the RA about 25%. When the arrival rate exceeded 1.0, the MDQN algorithm's channel utilization rapidly rose to over 94%, while the DQN algorithm's channel utilization was around 80%, the SAA's about 70%, and the RA's about 65%. This indicated that the MDQN algorithm significantly outperformed other algorithms in terms of channel utilization, with an improvement of about 34% compared to the SAA and about 45% compared to the RA, demonstrating superior performance. Similarly, in the average and rate metric tests shown in Figure 10(b), a similar trend was observed. As the arrival rate increased, the average and rate metrics of each algorithm also rose gradually. When the arrival rate exceeded 1.0, the MDQN algorithm's average and rate metrics reached over 25 bits/s/Hz, significantly surpassing other algorithms, with a notable improvement compared to the SAA and the RA. Additionally, compared to the DQN algorithm, the MDQN algorithm showed a more significant performance advantage at lower arrival rates. The comparison results of the transmission success rate of each algorithm task and the average reward of U are shown in Figure 11.



(a) Comparison outcomes of CU rates



(b) Comparison outcomes of average and rate

Figure 10: Robustness test results of different algorithms under arrival rates.

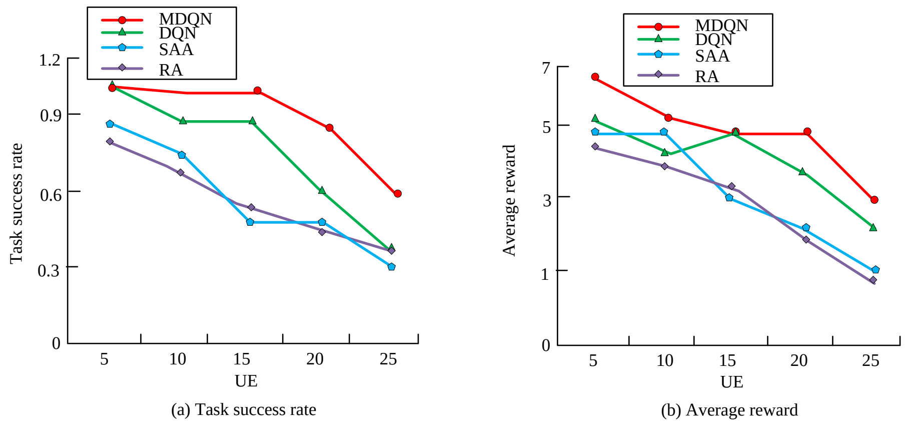


Figure 11: Comparison of task transmission success rates and average rewards on mobile devices for various algorithms.

Figure 11 (a) indicates the comparison results of task transmission success rates for various algorithms, and Figure 11 (b) indicates the comparison results of average rewards for mobile devices for various algorithms. From Figure 11 (a), overall, as the number of mobile devices increased, the success rate of each algorithm task transmission decreased. Among them, the success rate of MDQN algorithm task transmission was superior to other comparative algorithms, and its success rate could reach over 95% when the number of mobile devices was less than 15. From figure 11 (b), as mobile device numbers rose, all algorithms saw reduced average rewards, but MDQN achieved the highest average reward, outperforming others. This decline stemmed from increased co-channel interference, degraded signal quality, higher transmission delays, energy consumption, and failure rates, which collectively undermined system consistency, reliability, and overall performance. The comparison results of the latency of each algorithm system are shown in Figure 12.

From Figure 12, as the system bandwidth increased, the latency of each algorithm decreased, and the MDQN algorithm performed the best, with low system latency and computational costs. The MDQN algorithm had the lowest system latency under different numbers of blockchain nodes, and could quickly learn better RSAL schemes. The RA had a high latency, about 0.5-0.9ms higher than MDQN. The MDQN algorithm had a low system latency, effectively reducing task processing time.

In conclusion, to fully ascertain the real-world validity of the MDQN algorithm in IoT big data processing and the optimal use of communication channels, a set of rigorous and in-depth simulation tests were conducted. The aim was to skillfully incorporate the MDQN algorithm into the new generation of intelligent IoT network systems. In the test, multiple data streams of different business types were simulated, and through the processing and analysis of these data, the CU rate was accurately calculated. The results are shown in Figure 13.

From Figure 13, as the number of iterations increased, the effective CU rates of the traditional and research schemes gradually increased and tended to stabilize. Among them, under the research scheme, when the iteration number was 70, the effective utilization rate (EUR) of the channel rapidly increased and tended to stabilize, reaching 100%. However, under the traditional scheme, the EUR of the channel was less than 40%. The results showed that the EUR of the channel under the research scheme was higher, proving the effectiveness of the research algorithm in wireless RSMG.

Finally, to evaluate the computational complexity of the MDQN algorithm, it was compared with the standard DQN. In the standard DQN, assuming a state space of size S, an action space of size A, and each layer of the neural network having n_i neurons (with i layers), the main computational complexity comes from the forward and backward propagation of the neural network, which is approximately $O\left(\sum_i n_i^2\right)$. The MDQN algorithm improves upon the standard DQN by adding action space compression and a priority experience replay strategy based on TD errors. While action space compression reduced the number of actions, it also incurred additional computational costs, estimated at $O(A)$. The priority experience replay strategy adds extra computation when sampling and updating the importance weights, estimated at $O(N)$ (with N being the size of the experience buffer). Overall, the MDQN algorithm has a computational complexity of $O\left(\sum_i n_i^2 + A + N\right)$, slightly higher than the standard DQN but within an acceptable range, with a significant performance improvement.

To investigate the performance of MDQN in larger and more complex IIoT networks, experiments were conducted in a network comprising 300 devices. The algorithms DQN, SAA, and RA were compared, with evaluation metrics including channel utilization, system and rate, and system delay. Each metric was tested 10 times independently, with the average values and 95% confidence intervals calculated. Additionally, a significance test was performed ($p<0.05$ indicates a significant difference). The experimental results are presented in Table 5.

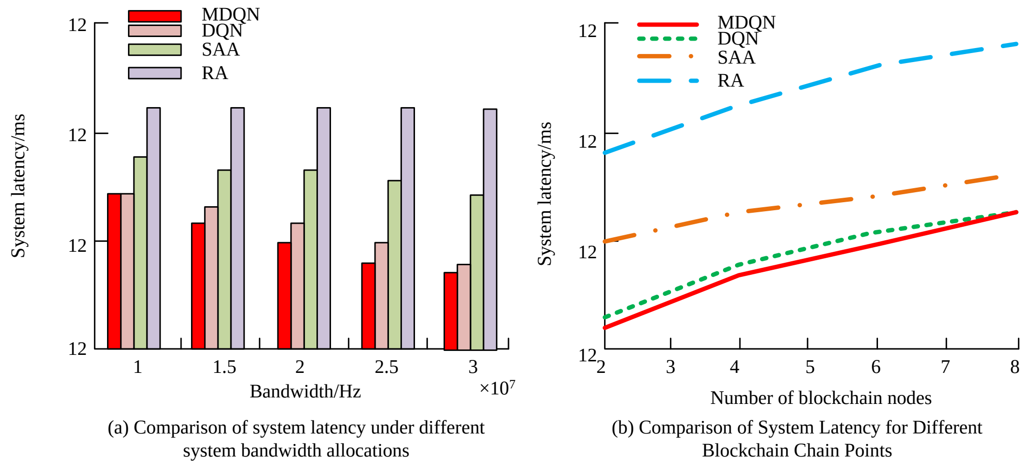


Figure 12: Comparison of time delay results of various algorithm systems.

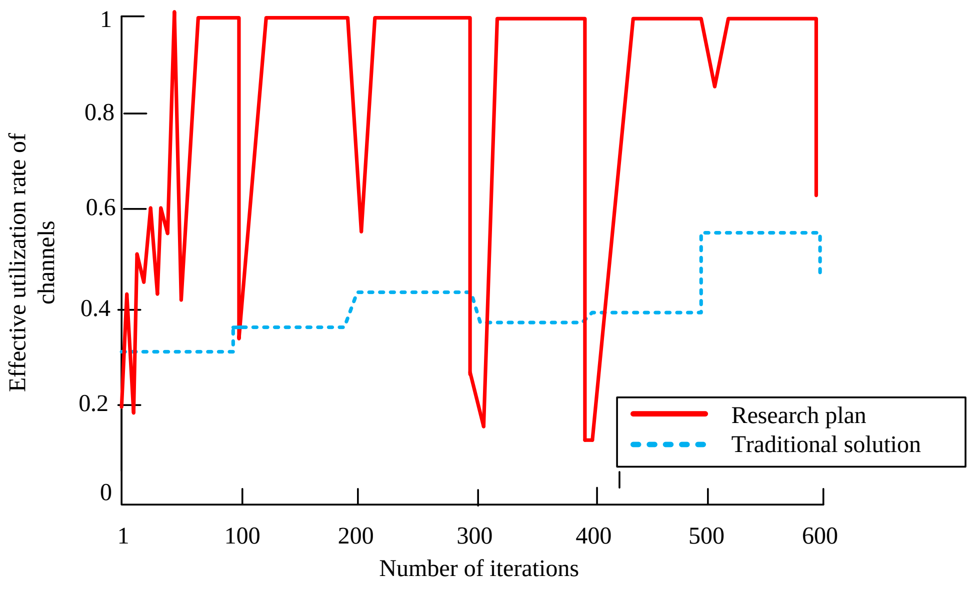


Figure 13: Changes in CU curve in random environment.

Table 5: MDQN performance in large-scale IIoT networks.

Algorithm	Channel utilization (mean ± 95% confidence interval)	System and rate (bits/s/Hz) (mean ± 95% confidence interval)	System delay (ms) (mean ± 95% confidence interval)	Significance test (compared with MDQN)
MDQN	91.5%±1.2%	23.2±0.8	0.27±0.02	-
DQN	84.3%±1.5%	19.8±1.0	0.34±0.03	*p*<0.05
SAA	68.7%±2.0%	10.5±1.2	0.43±0.04	*p*<0.05
RA	59.2%±1.8%	7.8±0.9	0.51±0.05	*p*<0.05

As shown in Table 5, the MDQN algorithm maintained excellent performance in larger-scale IIoT networks, achieving a channel utilization rate of 91.5% (95% confidence interval: 90.3%-92.7%), a system and rate of 23.2bits/s/Hz (95% confidence interval: 22.4-24.0bits/s/Hz), and a system delay of only 0.27ms (95% confidence interval: 0.25-0.29ms). Compared with other algorithms, the MDQN algorithm showed significant differences (*p*<0.05), demonstrating its strong generalization and scalability in more complex network environments.

The experiments showed MDQN surpassed PPO and A3C in channel utilization and system rate. MDQN's enhanced DQN structure, combined with action space compression and TD error-based prioritized experience replay, enabled efficient and accurate resource allocation in discrete IIoT spaces. It converged faster, offering a practical edge in real-time IIoT applications requiring rapid network adaptation, unlike PPO and A3C, which need more training iterations. This confirmed MDQN's effectiveness and superiority in IIoT settings.

4 Discussion

After presenting the experimental results, it is crucial to critically compare the performance of the proposed MDQN algorithm with the methods reviewed in the Related Work section. In terms of convergence speed, MDQN demonstrated a significant advantage over the traditional DQN algorithm. As shown in the training effect figures, MDQN achieved a stable state with fewer training iterations, while DQN required more iterations to converge. This is consistent with the findings of Naderializadeh N et al. [8], who also employed deep reinforcement learning for resource management but did not incorporate the improvements in action space compression and prioritized experience replay as in MDQN.

In terms of accuracy, MDQN achieved higher channel utilization and system sum rate compared to SAA and RA algorithms. This is because MDQN can better adapt to the dynamic IIoT environment by learning from historical experiences and making more informed resource allocation decisions. Shen Y et al. [10] used graph neural networks for radio resource management, which also showed good performance in large-scale scenarios. However, MDQN's approach of combining Markov decision processes with improved DQN provides a more flexible and efficient solution for IIoT networks.

Regarding computational efficiency, MDQN's action space compression technique reduced the computational complexity, making it more suitable for resource-constrained IIoT devices. Although it introduced some additional computational overhead for prioritized experience replay, the overall performance gain outweighed this cost. In terms of robustness, MDQN performed well under different network conditions, such as varying arrival rates and system bandwidths. This is in contrast to some of the methods reviewed, which may struggle to adapt to changing network environments. For example, the whale optimization algorithm proposed by Pham Q V et al. [9] focused on energy efficiency and spectrum efficiency trade-offs but may not be as robust in handling dynamic network loads as MDQN.

5 Conclusion

A wireless RSMG algorithm based on MDQN was proposed for IIoT systems containing multiple UEs and multiple spectrum resources, which combined MAC frame structure, spatial compression strategy, and TD-based priority experience replay strategy. The outcomes showed that the MDQN algorithm had substantially raised performance after training, surpassing RA and SAA, achieving high CU and system rate, and converging faster than DQN. When the reach rate was greater than 1.0, the channel rate and average sum rate of the MDQN algorithm reached 94% and 25bits/s/Hz respectively, significantly better than the compared algorithms, and more than twice as good as the SAA and RA.As the system bandwidth increased, the latency of each algorithm gradually decreased. Among them, the performance of the MDQN algorithm was superior to the compared algorithms, with lower system latency and computational costs, and the lowest system latency under different numbers of blockchain nodes. However, this study proposes that wireless RSMG algorithms have a high demand for computing resources due to their combination of multiple structures. To address these issues, lightweight network models will be explored in the future to reduce computational resource requirements. Meanwhile, algorithm performance can be improved through reasonable data preprocessing and enhancement.

The proposed algorithm excelled in simulations but faces real-world deployment challenges, primarily due to high computational demands from its integrated structures, which edge devices with limited power may struggle to handle. To address this, lightweighting through model compression and code optimization is essential to reduce computational complexity and enhance execution efficiency. Additionally, energy consumption must be managed to prevent excessive battery drain during operation.

6 Funding

The research is supported by: Liaoning Province Education Science 14th Five Year Plan Project: Research on the Integration System of Innovation and Entrepreneurship Education and Professional Education in Applied Undergraduate Universities (No. JG21EB039); Research Project on Adult Continuing Education in the 14th Five Year Plan of China Adult Education Association: Innovative Research on Cultivating New Vocational Farmers through Adult Continuing Education under the Rural Revitalization Strategy (No. 2023-621Y).

7 References

[1] Fatima Hussain, Syed Ali Hassan, Rasheed Hussain, and Ekram Hossain. Machine learning for resource management in cellular and IoT networks: potentials, current solutions, and open challenges. IEEE Communications Surveys & Tutorials, 22(2):1251-1275, 2020.https://doi.org/10.1109/COMST.2020.2964534

[2] Liangkun Yu, Rana Albelaihi, Xiang Sun, Nirwan Ansari, and Michael Devetsikiotis. Jointly optimizing client selection and resource management in wireless federated learning for internet of things. IEEE Internet of Things Journal, 9(6):4385-4395, 2021.https://doi.org/10.1109/JIOT.2021.3103715

[3] Wen Wu, Mushu Li, Kaige Qu, Conghao Zhou, Xuemin Shen, Weihua Zhuang, Xu Li, and Weisen Shi. Split learning over wireless networks: Parallel design and resource management. IEEE Journal on Selected Areas in Communications, 41(4): 1051-

1066, 2023. https://doi.org/10.1109/JSAC.2023.3242704

[4] Ramkumar Jayaraman, Baskar Manickam, Suresh Annamalai, Manoj Kumar, Ashutosh Mishra, and Rakesh Shrestha. Effective resource allocation technique to improve QoS in 5G wireless network. Electronics, 12(2): 451-469, 2023. https://doi.org/10.3390/electronics12020451

[5] Anurag Thantharate and Cory Bear. ADAPTIVE6G: Adaptive resource management for network slicing architectures in current 5G and future 6G systems. Journal of Network and Systems Management, 31(1): 9-32, 2023. https://doi.org/10.1007/s10922-022-09693-1

[6] Junhui Zhao, Yiwen Nie, Huan Zhang, and F. Richard Yu. A UAV-aided vehicular integrated platooning network for heterogeneous resource management. IEEE Transactions on Green Communications and Networking, 7(1): 512-521, 2023. https://doi.org/10.1109/TGCN.2023.3234588

[7] Rathinaraja Jeyaraj, Anandkumar Balasubramaniam, Ajay Kumara M.A., Nadra Guizani, and Anand Paul. Resource management in cloud and cloud-influenced technologies for internet of things applications," ACM Computing Surveys, 55(12): 1-37, 2023. https://doi.org/10.1145/3571729

[8] Navid Naderializadeh, Jaroslaw J. Sydir, Meryem Simsek, and Hosein Nikopour. Resource management in wireless networks via multi-agent deep reinforcement learning. IEEE Transactions on Wireless Communications, 20(6): 3507-3523, 2021. https://doi.org/10.1109/TWC.2021.3051163

[9] Quoc-Viet Pham, Seyedali Mirjalili, Neeraj Kumar, Mamoun Alazab, and Won-Joo Hwang. Whale optimization algorithm with applications to resource allocation in wireless networks. IEEE Transactions on Vehicular Technology, 69(4): 4285-4297, 2020. https://doi.org/10.1109/TVT.2020.2973294

[10] Yifei Shen, Yuanming Shi, Jun Zhang, and Khaled B. Letaief. Graph neural networks for scalable radio resource management: Architecture design and theoretical analysis. IEEE Journal on Selected Areas in Communications, 39(1): 101-115, 2020. https://doi.org/10.1109/JSAC.2020.3036965

[11] Amin Mohajer, Farid Sorouri, A. Mirzaei, A. Ziaeddini, K. Jalali Rad, and Maryam Bavaghar. Energy-aware hierarchical resource management and backhaul traffic optimization in heterogeneous cellular networks. IEEE Systems Journal, 16(4): 5188-5199, 2022. https://doi.org/10.1109/JSYST.2022.3154162

[12] Helin Yang, Jun Zhao, Zehui Xiong, Kwok-Yan Lam, Sumei Sun, and Liang Xiao. Privacy-preserving federated learning for UAV-enabled networks: Learning-based joint scheduling and resource management. IEEE Journal on Selected Areas in Communications, 39(10): 3144-3159, 2021. https://doi.org/10.1109/JSAC.2021.3088655

[13] Ying Chen, Zhiyong Liu, Yongchao Zhang, Yuan Wu, Xin Chen, and Lian Zhao. Deep reinforcement learning-based dynamic resource management for mobile edge computing in industrial internet of things. IEEE Transactions on Industrial Informatics, 17(7): 4925-4934, 2020. https://doi.org/10.1109/TII.2020.3028963

[14] Huimin Lu, Yin Zhang, Yujie Li, Chi Jiang, and Haider Abbas. User-oriented virtual mobile network resource management for vehicle communications. IEEE Transactions on Intelligent Transportation Systems, 22(6): 3521-3532, 2020. https://doi.org/10.1109/TITS.2020.2991766

[15] Jinjin Chao and Mengtian Jiao. Network spectrum resource allocation and optimization based on deep learning and TRDM. Informatica, 49(13):46-53, 2025. https://doi.org/10.31449/inf.v49i13.7374

[16] Qiao Qi, Xiaoming Chen, Caijun Zhong, Chau Yuen, and Zhaoyang Zhang. Deep learning-based design of uplink integrated sensing and communication. IEEE Transactions on Wireless Communications, 23(9): 10639-10652, 2024. https://doi.org/10.1109/TWC.2024.3373797

[17] Athanasios Karapantelakis, Pegah Alizadeh, Abdulrahman Alabassi, Kaushik Dey, and Alexandros Nikou. Generative AI in mobile networks: a survey. Annals of Telecommunications, 79(1): 15-33, 2024. https://doi.org/10.1007/s12243-023-00980-9

[18] Jiayin Wang, Yafeng Wang, Peng Cheng, Kan Yu, and Wei Xiang. DDPG-based joint resource management for latency minimization in NOMA-MEC networks. IEEE Communications Letters, 27(7): 1814-1818, 2023. https://doi.org/10.1109/LCOMM.2023.3266931

[19] Sadia Islam Nilima, Md Khokan Bhuyan, Md Kamruzzaman, Jahanara Akter, Rakibul Hasan, and Fatema Tuz Johora. Optimizing resource management for IoT devices in constrained environments. Journal of Computer and Communications, 12(8): 81-98, 2024. https://doi.org/10.4236/jcc.2024.128005

[20] Lei Liu, Jie Feng, Xuanyu Mu, Qingqi Pei, Dapeng Lan, and Ming Xiao. Asynchronous deep reinforcement learning for collaborative task computing and on-demand resource allocation in vehicular edge computing. IEEE Transactions on Intelligent Transportation Systems, 24(12): 15513-15526, 2023. https://doi.org/10.1109/TITS.2023.3249745

[21] Andy E. Williams. Human-centric functional computing as an approach to human-like computation. Artificial Intelligence and Applications, 1(2): 118-137, 2023. https://doi.org/10.47852/bonviewAIA2202331

[22] Yujun Wang. Deep learning models in computer data mining for intrusion detection. Informatica, 47(4): 188-196, 2023. https://doi.org/10.31449/inf.v47i4.4942

[23] Bitan Banerjee, Robert C. Elliott, Witold A. Krzymień, and Mostafa Medra. Machine-learning-aided TDD massive MIMO downlink transmission for high-mobility multi-antenna users with partial uplink channel state information. IEEE Transactions

on Wireless Communications, 24(1): 101-117, 2024.
https://doi.org/10.1109/TWC.2024.3485128