energies

Article

An Energy-Efficient Cross-Layer Routing Protocol for Cognitive Radio Networks Using Apprenticeship Deep Reinforcement Learning

Yihang Du [1], Ying Xu [1,*], Lei Xue [1], Lijia Wang [2] and Fan Zhang [3]

1 Electronic Countermeasure Institute, National University of Defense Technology, Shushan District, Hefei 230000, China
2 CTTL-Terminals of China Academy of Telecommunication Research of MIIT, Haidian District, Beijing 100191, China
3 Science and Technology Research Bureau of AnHui XinHua University, Shushan District, Hefei 230000, China
* Correspondence: keyanchu@axhu.edu.cn; Tel.: +86-551-6592-7432

Received: 26 June 2019; Accepted: 18 July 2019; Published: 22 July 2019

Abstract: Deep reinforcement learning (DRL) has been successfully used for the joint routing and resource management in large-scale cognitive radio networks. However, it needs lots of interactions with the environment through trial and error, which results in large energy consumption and transmission delay. In this paper, an apprenticeship learning scheme is proposed for the energy-efficient cross-layer routing design. Firstly, to guarantee energy efficiency and compress huge action space, a novel concept called dynamic adjustment rating is introduced, which regulates transmit power efficiently with multi-level transition mechanism. On top of this, the Prioritized Memories Deep Q-learning from Demonstrations (PM-DQfD) is presented to speed up the convergence and reduce the memory occupation. Then the PM-DQfD is applied to the cross-layer routing design for power efficiency improvement and routing latency reduction. Simulation results confirm that the proposed method achieves higher energy efficiency, shorter routing latency and larger packet delivery ratio compared to traditional algorithms such as Cognitive Radio Q-routing (CRQ-routing), Prioritized Memories Deep Q-Network (PM-DQN), and Conjecture Based Multi-agent Q-learning Scheme (CBMQ).

Keywords: cognitive radio networks; energy efficiency; apprenticeship learning; dynamic adjustment rating; Prioritized Memories deep Q-learning from Demonstrations

1. Introduction

Spectrum shortage has been one of the major bottlenecks for the development of wireless communication systems [1]. As a promising solution to the frequency scarcity, Cognitive Radio (CR) is presented to allow Secondary Users (SUs) to exploit licensed spectrum opportunistically, which breaks the fixed frequency allocation regulation. Great developments have been achieved in some crucial CR technologies such as spectrum sensing, spectrum access and power allocation [2–4]; however, there has been little study on cross-layer routing in the research community. Routing plays a very important role in the overall performance of any networks [5]. Although the CR technique improves spectrum utilization efficiency, it raises significant challenges to route selection in distributed Cognitive Radio Networks (CRN) [6].

Firstly, spectrum allocation should be considered as selecting the next hop in routing [7]. In traditional wireless communication, all of the nodes share the same frequency bands, whereas in multi-hop CRN, each node may work with different available channel sets based on the uncertainty of Primary Users' (PUs') activities. Thus, both the geographic location and spectrum bands have an effect on the connectivity of network topology. Secondly, it is challenging to develop an efficient and robust routing protocol in multi-hop CRN due to the spectrum uncertainty feature and the unreliable characteristic of the wireless medium [8].

Conventional routing protocols are used to apply explicit and rigid programming to opportunistic routing design. In [9], the virtual coordinates were introduced to reflect both geographic distance and spectrum availability, and then the geographic routing scheme was developed to cut through the area with lower frequency occupancy. However, it did not consider the interference issue among secondary users. In [10], a novel routing metric was designed to estimate both the future spectrum availability and the average transmission time. Then, two routing schemes were proposed to reduce the probability of spectrum handoff and rerouting. Nevertheless, power adaptation was not taken into consideration, which affected its energy efficiency. The author in [11] presented an on-demand routing protocol for a cluster-based cognitive radio ad-hoc network. The network was divided into several clusters, and then the opportunistic routing was formulated as a weighted graph problem. However, the rendezvous algorithm for neighbor discovery remained to be studied. With transcendental knowledge such as network topology and channel characteristics, these works attempted to search a fixed path. This was not practical, owing to the distributed network architecture and uncertainty nature of CRN.

To be truly intelligent and cognitive, the cognitive radio system should have the ability to learn and reason [12]. In recent years, the development of Deep Reinforcement Learning (DRL) has attracted great attention and has been widely used in the decision-making and control field. To resolve the curse of dimensionality in the problem with large state space, the Deep Q-Network (DQN) was proposed in [13]. It applied neural network to approximating the Q-function and improved the generalization ability of the algorithm. Simultaneously, the bias was introduced, and it was more difficult to guarantee the convergence. Due to the one-step update framework in DQN, its learning efficiency was relatively high. Furthermore, Policy Gradient (PG) was able to parameterize the strategy and learn the strategy directly [14]. It had better convergence properties and was effective in the continuous action space. However, PG typically converged to a local rather than global optimum, and it was inefficient since the strategy was updated every learning episode. To combine the advantages of DQN and PG, the Deep Deterministic Policy Gradient (DDPG) was presented in [15], introducing two independent neural networks called actor and critic, respectively. While the actor performed the actions without the need for optimized value function, the critic criticized the actions taken by the actor and kept updating the value function [16]. DDPG adopted a one-step update framework instead of updating the strategy after the whole episode, which was more efficient than PG. However, the computational complexity of DDPG was sufficiently high, since it had to train two independent neural networks at the same time.

These DRL frameworks have been used for the cross-layer routing design in intelligent networks. In [17], an intelligent routing protocol based on DQN was presented. It allowed choosing the best data transmission paths according to the best criteria and based on the network status. The author in [18] imposed the DDPG to optimize the routing in software-defined networks with the DDPG Routing Optimization Mechanism (DROM). It simplified network operation and maintenance by enhancing the performance with black-box optimization in continuous time. [17,18] utilized the DRL approach to overcome the uncertainty of network topology, but they failed to consider energy efficiency in the network. The author in [19] developed a RL-based joint spectrum access and clustering scheme to improve idle channel sensing and reduce system energy expenditure. In [20], Prioritized Memories Deep Q-Network (PM-DQN) was developed to solve the route selection and channel access. It introduced a concept called responsibility rating to compress the huge action space and realize power adaptation for lower power consumption and calculation complexity. Nevertheless, the responsibility rating merely allowed single-level transition of the transmit power, which brought about inefficiency of power allocation. Furthermore, a single source node for generating data streams was considered in [20]. If multiple source nodes exist in the network, every node has to run the DRL for routing. In addition, all of these works adopted a self-learning scheme that required many interactions with the environment through trial and error. Since each interaction means the establishment of a communication link, this resulted in large communication overhead, extensive energy consumption and long exploration time. Some applications in CRN cannot afford the extensive latency and energy expenditure. If there exists any node that has learned a stable routing strategy, it is unnecessary for the newly joined node to learn from scratch. By using DQfD, interactions with the real environment decrease significantly with the guidance of empirical data from the expert node. This accelerates the learning process and reduces the trial-and-error cost, which is more compatible with applications of CRN.

Since most SUs rely on batteries for power supply, energy constraints have become an urgent problem to be resolved [21]. Therefore, improvement of energy efficiency is significant for the practical application of CR technology. Three methods are adopted to reduce the energy consumption of the network: Firstly, dynamic adjustment rating is introduced for efficient power adaptation. Secondly, the trial-and-error cost is reduced with expert's demonstration data, which leads to lower energy expenditure. Thirdly, energy utilization ratio is included as a part of utility function in the DRL. In this paper, we apply an apprenticeship learning scheme called DQfD to the cross-layer routing design in CRN. Initially, the data streams emerge from a source node and it achieves a stable routing strategy through self-learning. After that, new data streams emerge from another source node, and it learns the routing protocol from the empirical data of the previous source node, instead of learning from scratch by itself. The source node to be learned from is defined as the expert source node, and the source node that needs guidance is called the apprentice source node. Our characteristic work can be featured by the following:

i. As far as we know, this is the first work that applies DQfD to the large-scale CRN with a single-agent learning framework. We consider the scenario in which an expert source node already exists, and new data flows are generated from another source node. Simultaneously, a small number of relay nodes change their location, and the network topology has little change compared with the previous network scenario. In this case, DQfD is adopted to learn strategies from the expert source node, which speeds up the learning process and achieves better network performance than that of the expert source node.
ii. To resolve the inefficiency in power adaptation when utilizing responsibility rating, a novel concept called dynamic adjustment rating is introduced, which regulates transmit power adaptively by comparing single-hop latency with double thresholds. This enables the transmit power to transfer multiple levels at each time step, until rational transmit power is achieved, which accelerates power assignment and improves the system performance.
iii. To save the system memory and further improve the performance, PM-DQfD is proposed. This makes it possible to periodically erase the inessential self-generated data and outdated expert demonstrations to release replay memory. In addition, it reduces data redundancy and optimizes memory structure, which produces better performance. Simulation results show that the proposed method decreases routing delay and enhances energy efficiency, as well as network stability.

The rest of this paper is organized as follows: Section 2 describes the system model. The novel metric called dynamic adjustment rating is introduced in Section 3. Section 4 presents the PM-DQfD-based cross-layer routing scheme, which is followed by the simulation results in Section 5. Finally, Section 6 concludes the paper.

2. System Model

In this section, a multi-hop CRN is considered, which consists of M PUs and N SUs. The licensed spectrum bands are assigned to PUs, and each Primary User (PU) occupies the specific sub-band with regularity. SUs have no authority to utilize any frequency bands and opportunistically access channels only when PUs are inactive. The communication infrastructure is abstracted as a directed graph $G(N, L)$, where each vertex $i \in N$ denotes a Secondary User (SU) node in the network. The directed link $(i, j) \in L$ on the graph represents a communication link between the SU nodes n_i and n_j that exists when the nodes are within transmission range of each other and hold at least one common channel.

The architecture of the CRN scenario is shown in Figure 1. In the networking scenario, the multi-hop CRN coexists with two centralized PU networks. PUs communicate with the PU base station through direct transmission. SUs transmit data flows from the source to the destination node in a multi-hop manner via CR routers. As Figure 1 shows, there exists an expert source node which has learned stable routing policy. On this foundation, a new source node generates data streams, and the location of a small number of intermediate nodes changes simultaneously.

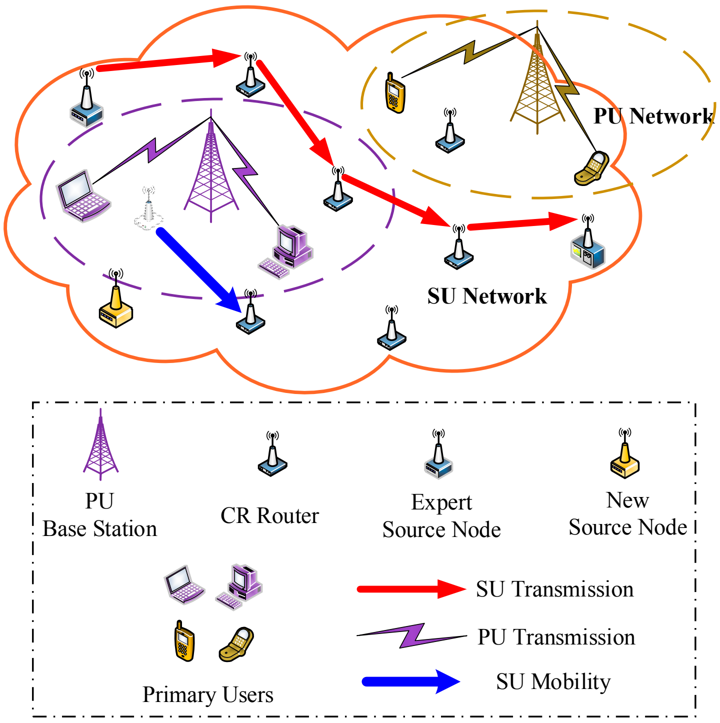


Figure 1. Multi-hop cognitive radio networking scenario.

The available spectrum bands comprise the Data Transmission Channel (DTC) and the Common Control Channel (CCC). SU utilizes DTC for data transmission and accesses it in the absence of the PU. CCC is used to exchange command information between SU nodes, and it can be used at any time. Furthermore, the PU's occupation is modeled as an independent and identical alternation between two stages, i.e., ON status, when the DTC is occupied, and OFF status, when the DTC is idle [22]. The probability density function of the OFF periods is demonstrated as follows:

$$g(\tau) = \begin{cases} \lambda e^{-\lambda\tau} & \tau \geq 0 \\ 0 & \tau < 0 \end{cases} \tag{1}$$

where λ is the mean OFF period of the PU. Hence, the probability that the PU arrives during the data transmission of SU n_i is calculated as

$$\begin{aligned} P_{collision} &= 1 - P(t \geq \tau_i) \\ &= 1 - \int_{\tau_i}^{\infty} g(t)dt \\ &= 1 - e^{-\lambda\tau_i} \end{aligned} \tag{2}$$

Here τ_i indicates the single-hop transmission latency of SU n_i and is calculated as:

$$\tau_i = \frac{S_{packet}}{B\log_2\left(1 + \frac{g_{ijc}p_i}{\xi + \phi_{ijc}^{PU}}\right)} \tag{3}$$

where S_{packet} represents data packet size, B denotes the bandwidth of DTC, g_{ijc} is the power gain between the node n_i and n_j, ϕ_{ijc}^{PU} represents the adjacent channel interference of PUs at node n_i, and ξ is the AWGN power.

To capture the energy efficiency of SUs, the Power Consumption Ratio (PCR) of SU n_i is introduced as follows:

$$\eta_i = \frac{p_i}{B\log_2\left(1 + \frac{g_{ijc}p_i}{\xi + \phi_{ijc}^{PU}}\right)} \tag{4}$$

PCR indicates power consumption as unit throughput is obtained. The parameters described in (4) are the same as in (3), which will not be repeated again.

3. Construction of Learning Framework

3.1. Dynamic Adjustment Rating

In the CRN scenarios, the selection criterion for the intermediate node stipulates that the relay SU lies within the transmission range and has common available channels. In addition, power allocation should be considered to enhance the energy utilization and prevent interference with PUs. In this case, the route selection, channel access, and power allocation constitutes the agent's action, leading to a huge action space.

Although DQN achieves excellent performance in problems with a large state space, its calculation complexity increases rapidly as the size of the action space becomes larger, which is a result of the characteristics of its update framework. The network parameters θ_i are updated through loss minimization using stochastic gradient descent:

$$L_i(\theta_i) = E\left[\left(r + \beta \max_{a\prime \in A} Q(s\prime, a\prime; \theta_i^-) - Q(s, a; \theta_i)\right)^2\right] \tag{5}$$

where r is the instantaneous return, β denotes the discounted factor, $Q(s,a)$ represents the Q-value at state s when choosing action a, and A is the action space of the agent. It can be seen from Equation (5) that the maximum computation is performed at every time-slot so that the calculation load will be heavy if the size of A is particularly large. Therefore, a novel concept called dynamic adjustment rating is introduced in this section to handle this problem.

A reasonable tradeoff between system energy expenditure and routing delay is demanded in CRN. It is reasonable that if the single-hop latency in the present time exceeds a higher threshold th_1, the SU node in the next hop will enhance its transmit power for lower transmission latency to maintain a reasonable and stable end-to-end delay. By contrast, if the current transmission latency is sufficiently short, i.e., less than a lower threshold th_2, the next intermediate node inclines in order to reduce its transmit power for system energy conservation. Otherwise, it is assumed that the current transmit power is being maintained within an appropriate range and should remain unchanged. According to this principle, the concept of responsibility rating was introduced in [20,23]. Nevertheless, as for the responsibility rating, the update framework is defined as follows:

$$\psi_i^t = \begin{cases} \psi_j^{t-1} + 1, & if\ \tau_j^{t-1} > th \\ \psi_j^{t-1} - 1, & if\ \tau_j^{t-1} \le th \end{cases} \tag{6}$$

where ψ_i^t is the responsibility rating of SU n_i at time step t, and it is a non-negative integer mapping to a level of transmit power. From Equation (6), we can see that if the single-hop delay is larger than the threshold th, ψ_i^t will increase by one unit in the next hop; otherwise, it will decrease by one unit. Since each responsibility rating ψ_i^t corresponds to a transmit power p_i, the transmit power only transfers one level for every timeslot. This leads that the process of power adaptation becoming quite long if the available power set is large, reducing power adjustment efficiency. Thus, a multi-level transition measurement called Dynamic Adjustment Rating (DAR) is presented. The DAR of SU n_i at timeslot t is given by:

$$DAR_i^t = \begin{cases} DAR_j^{t-1} + \varphi_i^t, & if\ \tau_j^{t-1} > th_1 \\ DAR_j^{t-1}, & if\ th_2 \le \tau_j^{t-1} \le th_1 \\ DAR_j^{t-1} - \mu_i^t, & if\ \tau_j^{t-1} < th_2 \end{cases} \tag{7}$$

Herein, DAR_j^{t-1} is the DAR of previous SU n_j at time step $t-1$, and τ_j^{t-1} denotes the single-hop latency of previous SU n_j at the last timeslot; DAR_i^t is a non-negative integer mapping to a level of transmit power, and two special cases are supplemented in Equations (8) and (9) when $DAR_j^{t-1} = 0$ and $DAR_j^{t-1} = \max\{DAR\}$:

$$DAR_i^t = \begin{cases} DAR_j^{t-1} + \varphi_i^t, & if\ DAR_j^{t-1} = 0\ and\ \tau_j^{t-1} > th_1 \\ DAR_j^{t-1}, & if\ DAR_j^{t-1} = 0\ and\ th_2 \le \tau_j^{t-1} \le th_1 \\ 0, & if\ DAR_j^{t-1} = 0\ and\ \tau_j^{t-1} < th_2 \end{cases} \tag{8}$$

$$DAR_i^t = \begin{cases} \max\{DAR\}, & if\ DAR_j^{t-1} = \max\{DAR\}\ and\ \tau_j^{t-1} > th_1 \\ DAR_j^{t-1}, & if\ DAR_j^{t-1} = \max\{DAR\}\ and\ th_2 \le \tau_j^{t-1} \le th_1 \\ DAR_j^{t-1} - \mu_i^t, & if\ DAR_j^{t-1} = \max\{DAR\}\ and\ \tau_j^{t-1} < th_2 \end{cases} \tag{9}$$

In Equation (7), φ_i^t is the increment index and is defined as:

$$\varphi_i^t = \begin{cases} \varphi_j^{t-1} + 1, & if\ \tau_j^{t-1} > th_1 \\ 0\,, & if\ \tau_j^{t-1} \le th_1 \end{cases} \tag{10}$$

That is, every time the transmission latency exceeds the higher threshold th_1, φ_i^t will increase by one unit; otherwise, it will return to 0. Conversely, the decrement index μ_i^t is given by:

$$\mu_i^t = \begin{cases} \mu_j^{t-1} + 1, & if\ \tau_j^{t-1} < th_2 \\ 0\,, & if\ \tau_j^{t-1} \ge th_2 \end{cases} \tag{11}$$

From Equations (7) and (10), it can be seen that the DAR will increase at accelerating levels each timeslot if transmission delay is consistently higher than the higher threshold th_1. On the contrary, the more successive timeslot single-hop latency falls below the lower threshold th_2, the larger the decrement of DAR will be. Thus, the DAR transfers multiple levels at each time step, which helps the transmit power of SU nodes reach a reasonable range rapidly and enhances the efficiency of power adaptation.

Every DAR maps to a value of transmit power p_i ($p_{\min} \le p_i \le p_{\max}$), and the mapping relation is given by:

$$p_i\left(DAR_i^t\right) = \left(1 - \frac{DAR_i^t}{|DAR_i|}\right)p_{\min} + \frac{DAR_i^t}{|DAR_i|}p_{\max} \tag{12}$$

where $|DAR_i|$ denotes the size of $\left\{DAR_i^t\right\}$. Consequently, with DAR, the transmit power is allocated efficiently to reach a reasonable tradeoff between energy efficiency and routing latency. Moreover, the DAR is able to transform the large action space problem to an issue with a huge state space, which reduces computation complexity and makes it compatible with DRL framework.

3.2. Construction of Learning Framework

The cross-layer routing problem is modeled as a Markov Decision Problem (MDP) [24], which is formulated as a tuple $M = \langle S, A, T, R \rangle$. Herein, S is the state space of the network environment,

A represents the action space; $T(s, a, s')$ is the state transition function. $R(s, a)$ specifies the reward received at state $s \in S$ when performing action $a \in A$.

Nevertheless, the transition function $T(s, a, s')$ is hard to obtain due to the dynamic and uncertainty characteristic of CRN. Considering the above problem, we introduce the model-free reinforcement learning to resolve the MDP without knowing the underlying game model [25]. The environment state, agent's available action and reward function are given in detail as follows.

3.2.1. Environment State

The state of network environment is defined as:

$$s_t = \left\{n_i, DAR_i^t\right\} \tag{13}$$

where $n_i \in \mathbf{N}$ represents the serial number of the present SU node, and $\mathbf{N}$ denotes the set of SU nodes; DAR_i^t is the dynamic adjustment rating of SU node n_i at timeslot t.

From the state definition in Equation (13), it is found that the present node n_i is merely decided by its previous relay node at the last timeslot, and DAR_i^t relies on DAR_j^{t-1}, as shown in Equation (7). Thus, the joint design problem satisfies the definition of Markov stochastic process. In addition, the learning episode terminates if n_i is the destination node, i.e., $\left\{n_{destination}, DAR_i^t\right\}$ is the terminal state in MDP.

3.2.2. Agent's Action

The relay selection, channel access and power assignment need to be jointly considered in the energy-efficient cross-layer routing scheme. Then the agent's action at time step t is given by:

$$a_t = \left\{n_k, c_i, p_{DAR_i}\right\} \tag{14}$$

where $n_k \in \mathbf{N}$ is the index of the selected relay node, and $\mathbf{N}$ represents the set of SUs; $c_i \in C_i$ represents the operating channel of SU n_i, and C_i is SU n_i's available DTC set; p_{DAR_i} is SU n_i's transmit power corresponding to its DAR.

If the power assignment is treated as a part of the action, the sizes of the action space and the state space are $|N| \times |C_i| \times |P|$ and $|N|$, respectively. With DAR, the action space size transforms into $|N| \times |C_i| \times 1$, since p_{DAR_i} is entirely decided by DAR_i, while the size of state space rises to $|N| \times |DAR_i|$, which will be sufficiently huge if both the network scale and the power set size become large. That is, DAR converts a large action space into a huge state space to make it more compatible with the DRL framework.

3.2.3. Reward Function

To guarantee energy efficiency and accommodate the maximal latency requirement, we aim to minimize the energy consumption and routing delay in the cross-layer routing problem. The PCR and single-hop delay are included in the instantaneous reward function, which is formulated as:

$$r_t(s_t, a_t) = -\log_2\left(\kappa_1 \cdot \eta_t^i + \kappa_2 \cdot \tau_t^i\right) \tag{15}$$

where η_t^i and τ_t^i denotes the PCR and single-hop delay of SU n_i at timeslot t, which are defined in Equations (4) and (3), respectively. κ_1 and κ_2 are coefficients that adjust the weighting between power expenditure and single-hop latency.

4. PM-DQfD-Based Energy-Efficient Joint Design Scheme

To accelerate the learning procedure, the DQfD-based apprenticeship learning scheme is adopted. The background of natural DQfD is introduced in Section 4.1. To reduce the memory occupation and further improve the system performance, PM-DQfD is presented in Section 4.2. Then in Section 4.3, the cross-layer routing scheme based on PM-DQfD is developed for lower end-to-end latency and system energy consumption.

4.1. Background of Natural DQfD

We consider a scenario in which there exists an expert source node that has already learned stable strategies. On this basis, new data flows are derived from another source node, and a small number of intermediate nodes change their location in the network. The strategies of the expert source node cannot be fully adapted to the new network scenario for two reasons. Firstly, the topology has slight changes compared with the previous network. Secondly, if the new source node lies in a remote area of the network, the expert's strategies in the state of the new source node may be imperfect, since few data packets are relayed by it. However, it will take a long time and extensive energy if the self-learning scheme is adopted in this case [26,27]. Thus, DQfD is considered, which accelerates the learning process with the aid of the expert's demonstration data [28]. Unlike the general learning scheme, DQfD allows the new source node to learn from the mature experience of the expert source node. Therefore, it performs well in tasks at the beginning of the learning process. The key link for DQfD is the pre-training stage. The agent utilizes the expert demonstrations to acquire relatively effective strategies. After pre-training with the demonstration data, interactions with the real environment are reduced, which decreases the cost of trial-and-error. Therefore, both the energy expenditure and the learning period decrease significantly.

In the initial stage, the expert source node learns the routing strategies using a self-learning framework called DQN. For the expert source node, the environment state is defined as $s_t = \left\{n_i, \psi_i^t\right\}$ and the action is given by $a_t = \left\{n_k, c_i, p_{\psi_i}\right\}$, where ψ_i^t is the responsibility rating of SU n_i defined in Equation (6), p_{ψ_i} is the transmit power of SU n_i corresponding to its responsibility rating, and the meaning of other parameters is the same as that of Equations (13) and (14). In addition, the reward function of the expert source node is the same as Equation (15). As shown in the learning framework depicted in Figure 2, the agent in the expert source node chooses an action a_t comprising route selection, channel access and power allocation, and then it observes the new state s_{t+1} and obtains the instantaneous reward r_t from the environment. The agent integrates them into the demonstration data $e_t = (s_t, a_t, r_t, s_{t+1})$ and stores this in the expert's replay memory. If the reward r_t is larger than a pre-defined threshold, the transition e_t will be stored in the demo buffer of the expert node as demonstration data. At a later point, these demonstration data will be transmitted to the apprentice source node through CCC. Back in the training step of the expert source node, in every timeslot, a mini-batch of transitions are sampled from the replay memory to calculate the loss function and update the network parameters based on gradient descent. At the next time step, the agent selects the action according to the Q-function approximated by the updated neural network. Finally, the expert source node gradually learns the optical strategy of cross-layer routing.

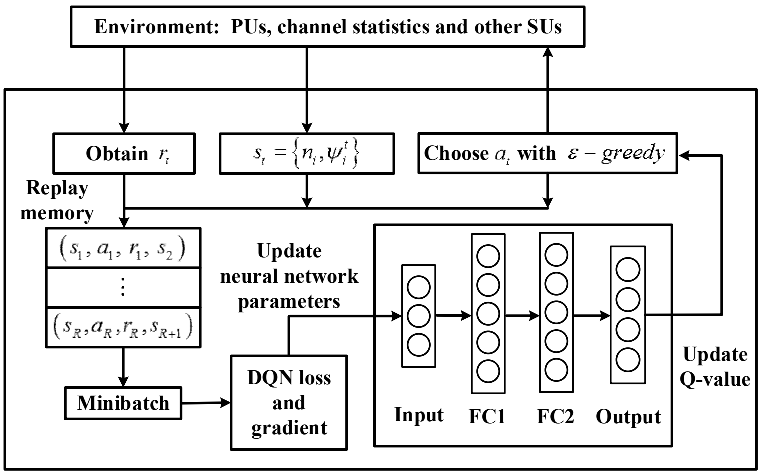

Environment: PUs, channel statistics and other SUs
Obtain r_t
$s_t = \{n_i, \psi_i^t\}$
Choose a_t with $\varepsilon - greedy$
Replay memory
(s_1, a_1, r_1, s_2)
⋮
(s_R, a_R, r_R, s_{R+1})
Update neural network parameters
Minibatch
DQN loss and gradient
Input
FC1
FC2
Output
Update Q-value

Figure 2. Implementation of memory optimization for the replay buffer.

In the pre-training stage, the loss function consists of the Bellman update loss, the supervised loss, and the L2 regularization loss [29]. The Bellman update loss $L_B(Q)$ guarantees that the Bellman equation is satisfied, and Q-value update can adopt online TD-learning in the later period. The supervised loss plays an important role in the effectiveness of pre-training. This is because the demonstration data only covers a limited state space and cannot traverse all possible actions. Many state-action pairs did not appear in the pre-training stage, so there is no data to ensure the corresponding Q-function converges to reasonable values [29]. To avoid this, the supervised loss is introduced as follows:

$$L_S(Q) = \max_{a \in A}[Q(s,a) + l(s,a_E,a)] - Q(s,a_E) \tag{16}$$

where a_E is the action chosen in the demonstration data, and $l(s,a_E,a)$ is a margin function given by:

$$l(s,a_E,a) = \begin{cases} 0 & ,a = a_E \\ C & ,else \end{cases} \tag{17}$$

where C is a large positive constant. It is found that if a_E achieves the maximum Q-function among all actions, the supervised loss $L_S(Q) = 0$; otherwise, $L_S(Q) = Q(s,a) + C - Q(s,a_E) > 0$. This ensures the parameters of the neural network update towards the distribution of the expert's Q-function, which reaches its maximum value when performing a_E. In addition, to avoid over-fitting the L2 regularization loss, $L_{L2}(Q)$ is taken into consideration. Consequently, the overall loss function is represented as follows:

$$L(Q) = L_B(Q) + \gamma_1 L_S(Q) + \gamma_2 L_{L2}(Q) \tag{18}$$

where γ_1 and γ_2 are parameters that regulate the weight between $L_S(Q)$ and $L_{L2}(Q)$.

Once pre-training is completed, the neural network has been trained well without trial-and-error cost. Then the agent enters the self-learning stage, in which it will continue to improve the decision-making level through interactions with the real environment. During the self-learning process, the agent selects the action $a_t = \{n_k, c_i, p_{DAR_i}\}$ according to the strategy learned in the pre-training phrase, which consists of route selection, channel access and power allocation. Then the agent observes the next state of the environment s_{t+1} and obtains the reward r_t. The self-generated data is formulated as $d^{self} = (s_t, a_t, r_t, s_{t+1})$ and stored in an independent replay buffer D^{replay}, which is updated every timeslot. Meanwhile, the demonstration data stored in D^{demo} remains static. After the amount of self-generated data reaches a certain value, the agent starts sampling a mini-batch of data randomly from D^{demo} and D^{replay} in a specific proportion, which is used to calculate the loss function and update the network parameters by gradient descent. Then the agent will be more intelligent when choosing the next action a_{t+1}, and the routing strategy will be more efficient. In addition, it is important to note that the Bellman update loss, as well as the supervised loss, is applied for the demonstration data; while for the self-generated data, only the Bellman update loss is applied.

Moreover, if the self-learning process is finished without acceptable results, the agent cannot learn the optimal routing strategy, which leads to lower system energy efficiency and longer end-to-end delay. There are two solutions to this situation. Firstly, we can increase the training steps κ in the pre-training phrase, which leads the agent to obtain a better strategy after pre-training. Secondly, the agent can continue training its neural network using expert demonstrations after the self-learning phrase until its performance is within an acceptable range. The performance of the second method will be better if the demonstration data in the demo buffer is updated using the expert source node after pre-training.

4.2. Prioritized Memories Deep Q-Learning from Demonstrations

Prioritized experience replay can be adopted to sample high-value data more frequently for performance improvement. However, the agent still needs to maintain both of the large replay buffer D^{replay} and large demo buffer D^{demo}, which increases the storage occupancy. Therefore, consideration about how to release storage space and utilize the data efficiently has become a crucial issue.

To address the problem discussed above, the Prioritized Memories Deep Q-learning from Demonstrations (PM-DQfD) is presented, which optimizes the memory structure of both demo and replay buffer. It erases the ineffective transitions and biases the memory contents to where the data value remains high. In DRL, transition's TD-error δ is defined as follows:

$$\delta = r + \beta \max_{a\prime \in A} Q(s\prime, a\prime; \theta_i^-) - Q(s, a; \theta_i) \tag{19}$$

It measures the potential for performance improvement when using the data for training [30]. So TD-error is adopted as an effective index to evaluate the value of transitions. To be specific, the transitions whose TD-errors are lower than the threshold will be swept off the replay buffer D^{replay}, and expert demonstration with the lowest TD-error will be erased from the demo buffer D^{demo}.

In Figure 3, we illustrate the implementation of memory optimization for the replay buffer D^{replay}. The transition at timeslot t is formulated as $e_t = (s_t,\ a_t\ , r_t\ , s'_t,\ \delta_t)$, where δ_t denotes the TD-error of experience. For the convenience of memory optimization, δ_t is set as ∞ before e_t is replayed. The real TD-error will be rewritten after the transition is sampled. Firstly, the learning process does not start before the transitions' amount is more than half the size of the replay buffer. As shown in Figure 3a, transitions are stored in sequence and number of transitions grows. In the second phase, the agent begins learning, and the quantity of transitions is below $a\%$ $(a > 50)$ of the buffer. We can see from Figure 3b that every timeslot a mini-batch of transitions is replayed, while the total number of transitions simultaneously increases. Then we define the Memory Erasing Threshold (MET) at timeslot t as:

$$MET_t = rand(0, \varepsilon) * \frac{1}{R_t} \sum\nolimits_{n=0}^{R_t} \delta_n \tag{20}$$

where R_t is the number of replayed transitions at timeslot t, and ε is a certain value between 0 and 1. If the quantity of replayed transitions exceeds $a\%$ of the buffer size, the transitions whose TD-errors are lower than MET will be swept off replay buffer D^{replay} until the quantity is less than $b\%$ $(b < a)$ of the buffer size. As shown in Figure 3c, the number of transitions decreases sharply in this period, and only transitions with sufficiently high TD-error are preserved. The second and the third step are executed alternately until the learning procedure converges.

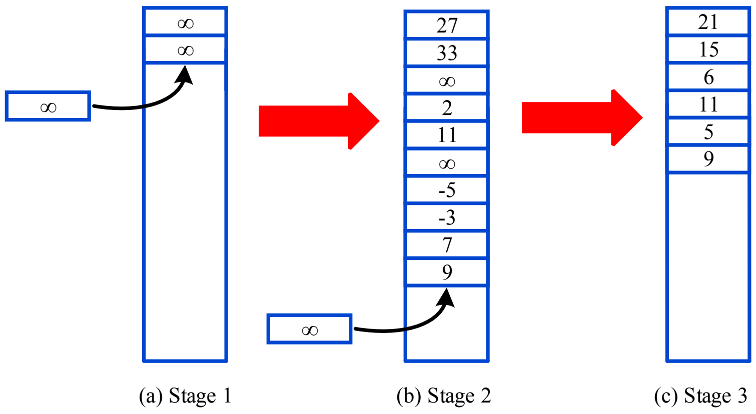


Figure 3. Implementation of memory optimization for the replay buffer.

On the other hand, the expert demonstrations may be subsequently outdated due to the dynamic nature of the radio environment. Thus, the reliability of the expert demonstrations is reduced for the agent, and it is unnecessary to maintain as much demonstration data as the initial phrase. Based on this, we erase q transitions with the lowest TD-error from the demo buffer D^{demo} every time when the learning episode is finished. When the amount of data decreases to a particular value S, we stop erasing demonstrations, and the number of demonstrations remains unchanged after that.

Memory structure optimization regularly erases worthless transitions, which saves system memory and efficiently utilizes the data. PM-DQfD is shown in detail as Algorithm 1.

Algorithm 1 Prioritized Memories Deep Q-learning from Demonstrations

```
1: Initialize:
2:   Input replay memory $D^{replay} = \phi$, demonstration data set $D^{demo}$, replay period $P$,
     demonstration ratio $\upsilon$, number of pre-training steps $\kappa$ and memory size $R$.
3:   Initialize the Q-function and the target Q-function.
4: Pre-training:
5:     For steps $t \in \{1, 2, \ldots, \kappa\}$ Do
6:         Sample $n$ transitions randomly from $D^{demo}$.
7:         Compute the overall loss function.
8:         Perform gradient descent to update $\theta$.
9:     End For
10: Self-learning:
11:     For episode $\in \{1, 2, \ldots, T\}$ Do
12:         Initialize state $s_1$.
13:         For steps $t \in \{1, 2, \ldots\}$ Do
14:             Choose action $a_t$ based on policy $\pi^{\epsilon Q_\theta}$.
15:             Execute action $a_t$ and observe reward $r_t$, state $s_{t+1}$.
16:             Store transition $(s_t, a_t, r_t, s_{t+1}, \delta_t = \infty)$ in $D^{replay}$.
17:             Sample $n$ transitions randomly from $D^{demo} \cup D^{replay}$ with a fraction $\upsilon$
               of the samples from $D^{demo}$.
18:             Compute the overall loss function.
19:             Perform gradient descent to update $\theta$.
20:             Calculate MET according to (18).
21:             If $(R_t > a\% * N)$ $or$ $((R_t < R_{t-1})$ $and$ $(R_t \geq b\% * N))$
22:                 For $transition = 1, N$ Do
23:                     If $\delta_{transition} < MET_t$
24:                         Erase the transition from $D^{replay}$.
25:                 End For
26:             Every $P$ steps reset $\hat{Q} = Q$.
27:         End For
28:         If $\left|D^{demo}\right| > S$
29:             Erase $q$ transitions with the lowest $\delta_t$ from $D^{demo}$.
30:     End For
```

4.3. PM-DQfD-Based Energy-Efficient Cross-Layer Routing Scheme

In this section, the PM-DQfD is applied to the cross-layer routing design for improvement of power efficiency and reduction of transmission delay. To combine the single-agent DRL with wireless network environments, an effective mechanism is designed. In the protocol, only the source node needs to deploy the cognitive engine and the replay memory, which radically reduces the computation load and communication overhead. The implementation is demonstrated as follows:

The complete learning stage consists of three parallel steps: data transmission, action transmitting and transition back propagation. As shown in Figure 4, we take the learning process of the SU node n_i at stage t as an example. Firstly, after the data transmission of link (j,i), SU n_j generates the transition e_t of the link (j,i) and back propagates e_t to the source node n_0 via CCC. Meanwhile, the SU node n_i chooses the intermediate node n_k as its next relay node and starts transmitting data to n_k through DTC c_i according to the action selection at the previous stage $a_{t-1} = \left\{n_k, c_i, p_{DAR_i}\right\}$. During the data transmission, the agent in source node n_0 updates network parameters to θ_t by sampling a mini-batch of transitions. Then it selects action a_t of the next relay node n_k and sends it to n_k through the established route. The three parallel steps are executed hop by hop until the destination node successfully receives the data packet.

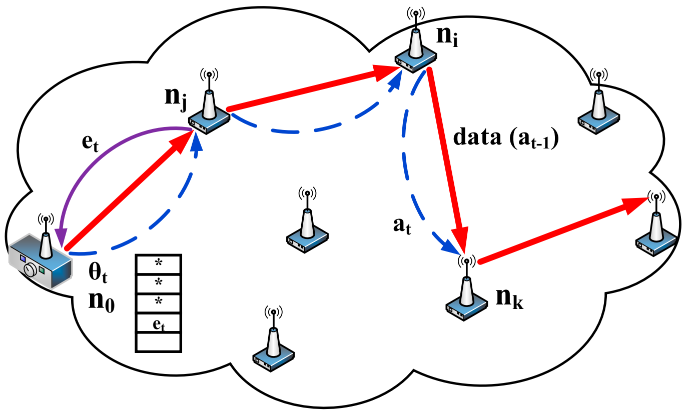


Figure 4. Implementation of PM-DQfD-based cross-layer routing scheme.

Furthermore, memory structure optimization is adopted in the joint design scheme. When the transition number is increased to a certain extent, it will release the storage space in order to reduce memory occupation. The memory saved can be used to store more data caches. For example, it is possible to store sensor data with higher precision for wireless sensor networks. Therefore, the PM-DQfD-based joint design algorithm is compatible with wireless environments in multi-hop CRN. Although the PM-DQfD-based protocol has many advantages over traditional schemes, a tradeoff is made between system performance and signaling overhead. The signaling overhead of the PM-DQfD-based cross-layer routing protocol consists of two parts. Firstly, the expert demonstrations are transmitted from the expert source node to the apprenticeship source node through the established route via CCC. Secondly, action transmission and transition back propagation are performed every hop in the single-agent-based routing mechanism. However, since the size of both transition and demonstration buffer is limited, the signaling overhead during the demonstration data transmission is in an acceptable range. In addition, the size of one action is sufficiently small compared to the data packet so that the signaling overhead caused by the action transmission and transition back propagation is relatively low. The interactions with the real world decrease despite these overheads, which is worthwhile for the significant decline of the power consumption and transmission latency.

5. Experiment and Evaluation

5.1. Simulation Setup

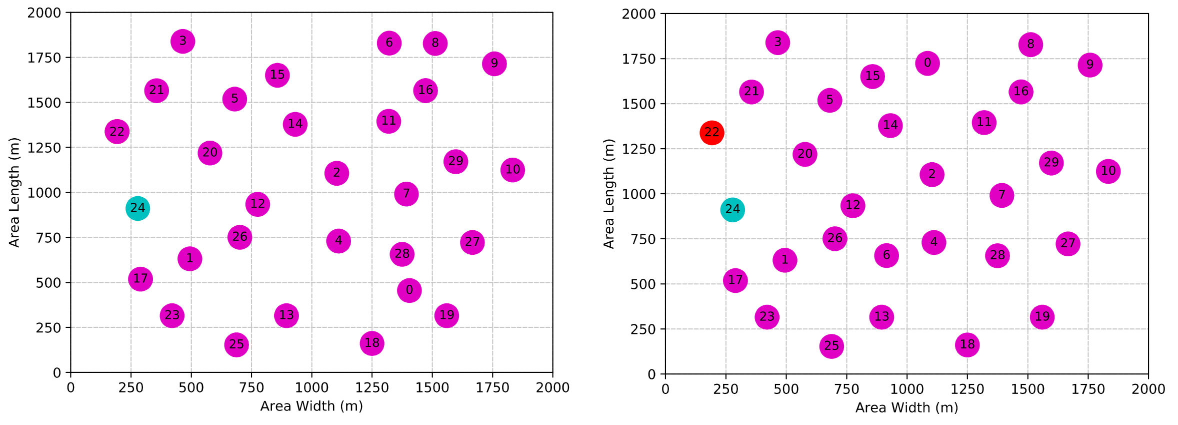
Through pre-training with the expert demonstrations, the agent passes over the initial stage and its neural network is trained without trial-and-error cost. Furthermore, memory optimization helps the agent exploit the data more efficiently and further improves its performance. All of this means that the PM-DQfD-based scheme is able to achieve faster convergence speed, less memory occupancy, higher energy efficiency and lower routing latency, which is vital to effectiveness and reliability in wireless communication. Thus, the approach is especially suitable for energy-constrained and delay-limited applications in CRN.

In this section, the performance of the PM-DQfD-based scheme is assessed. The results of the proposed scheme are compared with traditional schemes such as i) Deep Q-Network (DQN) [31]; ii) Prioritized Memories Deep Q-Network (PM-DQN) [20]; iii) Natural DQfD [29]; iv) Conjecture Based Multi-agent Q-learning Scheme (CBMQ) [32]; and v) Cognitive Radio Q-routing (CRQ-routing) [33] with respect to effectiveness, robustness and learning speed. The simulation framework was built using Python (3.5.1, Google, Mountain View, CA, USA). Specifically, the network environment and learning algorithm were coded on the basis of the Networkx (2.3, Los Alamos National Laboratory, Los Alamos, NM, USA) and Tensorflow (1.4.0, Google, Mountain View, CA, USA), respectively. It was assumed that the expert source node applies DQN.

In addition, we consider a networking scenario comprising 30 SU nodes. The location of nodes is subject to a two-dimensional uniform distribution. It is assumed that the coordinate of a SU node is (x, y), and its probability density function is given by:

$$f(x,y) = \begin{cases} \frac{1}{A}, & (x,y) \in G \\ 0, & \text{otherwise} \end{cases} \tag{21}$$

where G is a bounded domain on the plane (G is a square area of 2000×2000 m^2 in our simulation process), and A is the size of bounded domain G. The network topology is shown in Figure 5. In Figure 5a, SU node n_{24} is the source node and it establishes route to the destination node n_{10} through DQN. In Figure 5b, SU node n_{22} becomes a newly joined source node, and it routes messages by learning the routing protocol from the expert source node n_{24}. Simultaneously, the relay node n_0 and n_6 have moved, and some changes take place in the network topology. Furthermore, there are 10 licensed channels available for SUs, and the set of transmit power comprises 40 levels: $\{100,\ 120,\ 140,\ \ldots,\ 1000\ mW\}$. The remaining parameters are demonstrated in Table 1.



(**a**) (**b**)

Figure 5. Network topology consisting of 30 SUs: (**a**) Route establishment by self-learning, and (**b**) route establishment by apprenticeship learning.

Table 1. Simulation parameters.

Parameter	Default Value
Link Gain	$g = \varepsilon G(r/r_0)^{-m}$ [34]
Available Spectrum	56 MHz–65 MHz
Bandwidth, B	1 MHz
AWGN power, ξ	10^{-7} mW
PU-to-SU interference, ϕ_{ijc}^{PU}	$\phi_{ijc}^{PU} \sim \left[10^{-7}, 10^{-6}\right)$ mW
Packet Size, S_{packet}	2×10^5 *bit*
Mean OFF Period of PU, λ	2 *ms*
SINR threshold, σ_{th}	−80 *dBm*
Higher Latency Threshold, th_1	16 *ms*
Lower Latency Threshold, th_2	12 *ms*
Weighting of the power consumption ratio, κ_1	100
Weighting of the single-hop delay, κ_2	1
Demonstration ratio, υ	0.1
Discount factor, β	0.99
Learning rate, α	0.001
Weighting of the supervised loss, γ_1	1
Weighting of L2 regularization loss, γ_2	10^{-5}
Pre-training step, κ	200
Replay period, P	300
Size of Mini-Batch, ρ	64
Upper Limit of Erasing, a	99
Lower Limit of Erasing, b	70
Erasing Parameter, ε	0.2
Final Size of Demo Buffer, S	1500
Erasing parameter of Demo Buffer, q	3
Replay buffer size, R	2000
Demonstration buffer size, R	2000

5.2. Simulation Results

Figure 6 shows the quantity variation trends of self-generated data and expert demonstrations in the self-learning stage. We find that the amount of demonstration data remains constant before 1000 time steps. After that, the agent starts self-training and the size of the demonstration buffer decreases until it reaches 1500. On the other hand, the quantity of self-generated data increases monotonically before 2000 time steps. Then the total amount of the self-generated data fluctuates between b% and a% of the replay buffer size as memory optimization is performed. The decrease of demonstration data and the dynamic change of the self-generated data save the storage space, which can be used for storing more cache data or sensor data with higher precision.

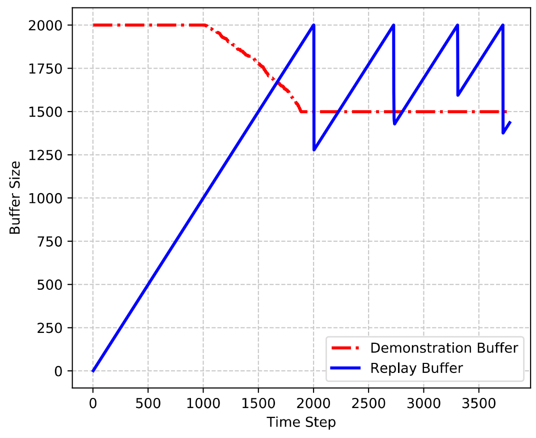


Figure 6. Quantity variation trends of self-generated data and demonstration data.

The data efficiency of the natural DQfD and PM-DQfD is investigated in Figure 7. Figure 7a presents the average TD-errorr of natural DQfD and PM-DQfD as they change with the learning timeslot. It can be seen that the average TD-error of both algorithms increases with timeslot, and the TD-error of PM-DQfD is significantly higher than that of DQfD. This is because data with low TD-error is swept off the demonstration buffer and the replay buffer when using PM-DQfD, which enhances the average TD-error. These results comply with common sense. In addition, the average TD-error of PM-DQfD surges as the self-generated data is erased. After that, the TD-error declines until it stabilizes, and the stable value of TD-error is larger than the value before the surge. The reason for this is that a large amount of high-quality data is retained after executing memory erasing, and it brings accumulation advantage for the subsequent phases. Figure 7b demonstrates the overall loss of the natural DQfD and PM-DQfD changing with time steps. It is observed that the overall loss of two algorithms shows a trend of fluctuating downward, and the loss of PM-DQfD is lower than that of DQfD. Moreover, the overall loss of PM-DQfD shows a sharp decline every time data erasing is performed. This is mainly because PM-DQfD periodically erases transitions with low TD-error, and the average TD-error of the remaining data gets larger. Data with larger TD-error make greater contributions to parameter training for neural network than other data. Thus, PM-DQfD causes lower loss and achieves better performance.

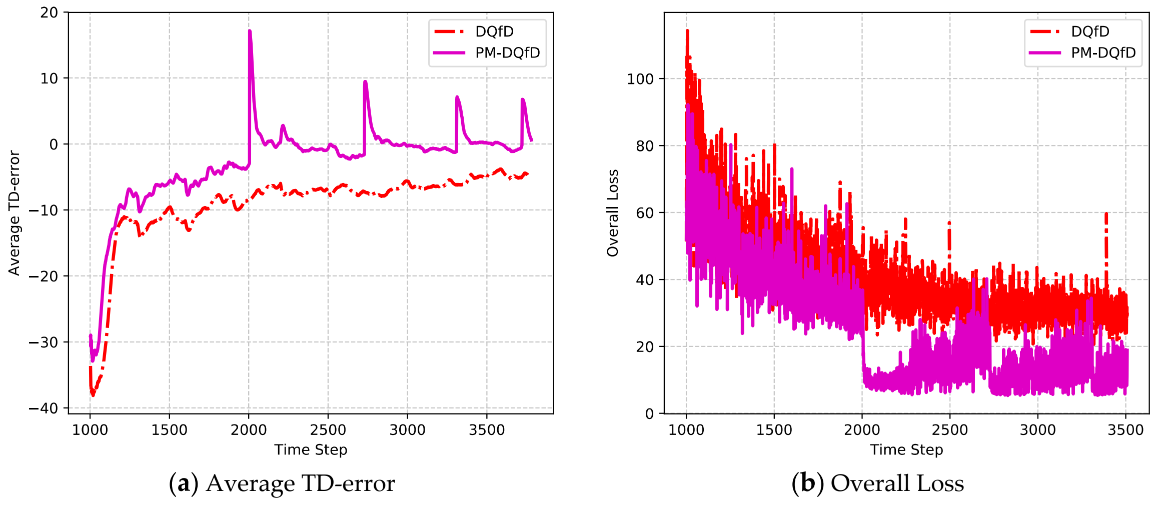


Figure 7. (**a**,**b**) Data efficiency of the natural DQfD and PM-DQfD versus time step.

Figure 8 demonstrates that with increasing learning episodes, the expected reward increases gradually for all algorithms. The red curve represents the reward of PM-DQfD with DAR. Other curves depict the performance of PM-DQfD, DQfD, PM-DQN, DQN, and CRQ-routing with responsibility rating, respectively. It is found that the expected return of DQfD-based schemes is generally higher than that of CRQ-routing and the two DQN-based algorithms. This is because the network is well trained after pre-training for the DQfD-based schemes, leading to more sophisticated strategies. In addition, the rewards of the two prioritized memories-based schemes, i.e., PM-DQfD and PM-DQN, are larger than those of natural DQfD and DQN, respectively. The reason for this is that the prioritized memory-based schemes periodically erase transitions with low TD-error, retaining high-quality data that has greater value for policy learning. Furthermore, better performance is achieved when the DAR is applied to PM-DQfD. Since the transmit power can be tuned to a reasonable value with fewer hops when applying DAR, lower energy consumption and end-to-end delay are obtained in each learning episode.

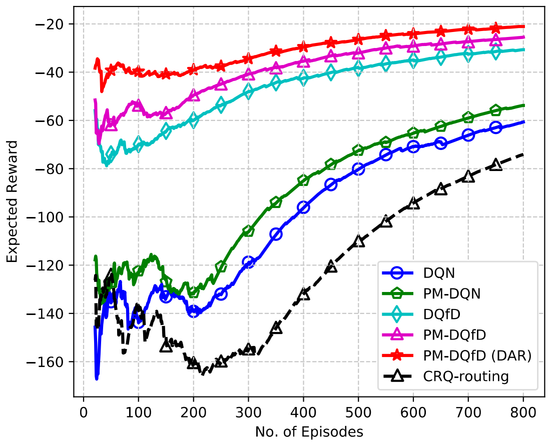
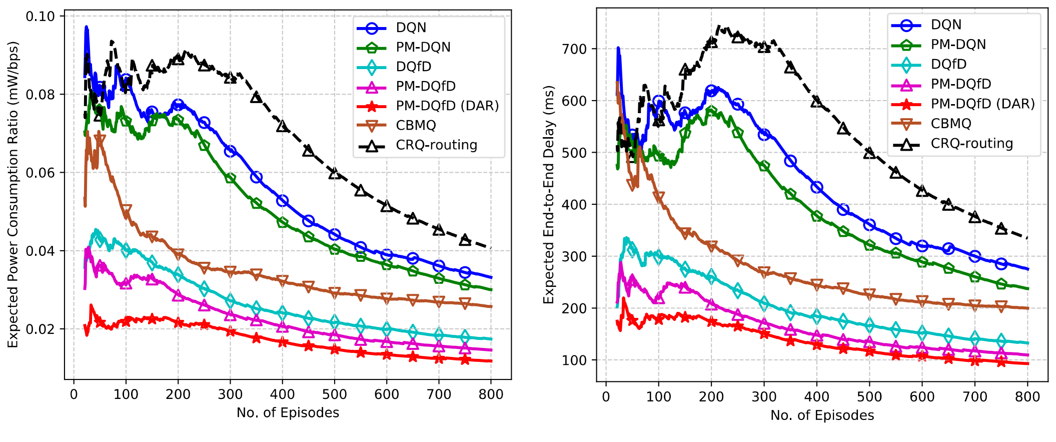


Figure 8. Expected reward versus number of episodes.

The effectiveness properties of the energy efficiency and routing latency are investigated in Figure 9. In Figure 9a, we depict the system power consumption ratio changing with the number of episodes. For all algorithms, system PCR experiences a gradual decline as the learning process goes on. From these curves, it can be concluded that the DQfD-based routing protocols achieve lower energy expenditure and faster learning speed than PM-DQN, DQN and CRQ-routing. Moreover, the single-agent framework, DQfD-based schemes have almost the same convergence speed as the multi-agent CBMQ scheme, and achieve better performance than it with regard to energy efficiency. This is because in DQfD-based schemes, the pre-training stage decreases the interactions with the real environment, which decreases trial-and-error cost as well as energy consumption. Furthermore, PM-DQfD holds an advantage over natural DQfD. This is because more valuable transitions are preserved with the memory structure optimization and redundancy reduction, which helps the agent achieve an energy-efficient routing protocol. In addition, since power adaptation is completed with fewer hops when applying DAR, the average PCR on the whole path decreases. Hence the system energy efficiency of DAR-based PM-DQfD is further improved compared with the responsibility rating-based schemes. In Figure 9b, we evaluate the trend of expected end-to-end delay changing with the learning episodes. It is similar to Figure 9a, since the weight of routing latency is equal to that of PCR when constructing the instantaneous reward in Equation (15). PM-DQfD with DAR achieves the shortest routing latency in all algorithms. In addition, it converges as fast as the multi-agent scheme CBMQ, which has a faster convergence speed than CRQ-routing and DQN-based protocols. No more expatiation here, since the reason for this is the same as in Figure 9a.



(**a**) Expected power consumption ratio (**b**) Expected end-to-end delay

Figure 9. (**a**,**b**) Effectiveness properties versus number of episodes.

In Figure 10, we investigate the average hops along the path changing with the number of learning rounds. It is found that DQfD-based schemes achieve lower average hops compared with other algorithms. Furthermore, the learning speed of DQfD-based schemes is the same as CBMQ, which converges much faster than CRQ-routing and two DQN-based algorithms. The reason for this is that pre-training with demonstration data accelerates the parameter-tuning of neural network in DQfD-based schemes. It is worth mentioning that the average number of hops of CBMQ are almost the same as that of PM-DQN. This is because the advantage of multi-agent schemes is the reduction in the power consumption and transmission delay of each hop, rather than achieving fewer hops along the path. Similarly, the prioritized memory-based protocols achieve fewer learning steps than the natural algorithms. This is due to the fact that data with high TD-error decreases blind exploration for relay nodes. However, applying DAR to PM-DQfD has no effect on reducing exploration steps in one episode. This is because the DAR can only produce a higher reward in each learning episode, but is invalid for route selection.

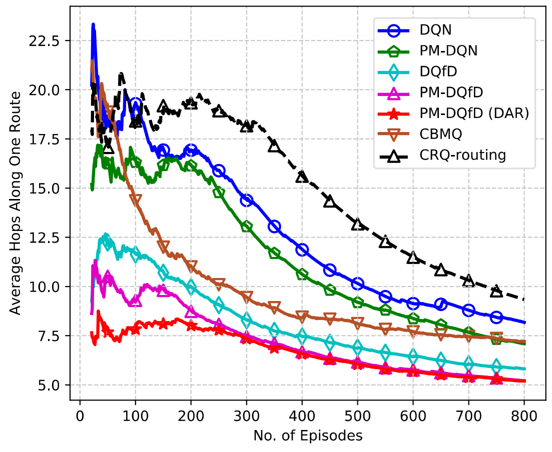


Figure 10. Average number of hops along the path versus number of episodes.

As shown in Figure 11, the Packet Delivery Ratio (PDR) increases for all algorithms with the increasing number of episodes. It can be observed that the PDR of DQfD-based schemes outperforms CBMQ, CRQ-routing and the two DQN-based methods. Among the DQfD-based methods, PM-DQfD with DAR achieves the highest PDR. Since PM-DQfD with DAR achieves the lowest end-to-end delay, as shown in Figure 9b, the portion of its packet delivery with a routing latency below the setting threshold exceeds that of other schemes. This illustrates that the PM-DQfD-based DAR is more robust and reliable compared to traditional algorithms.

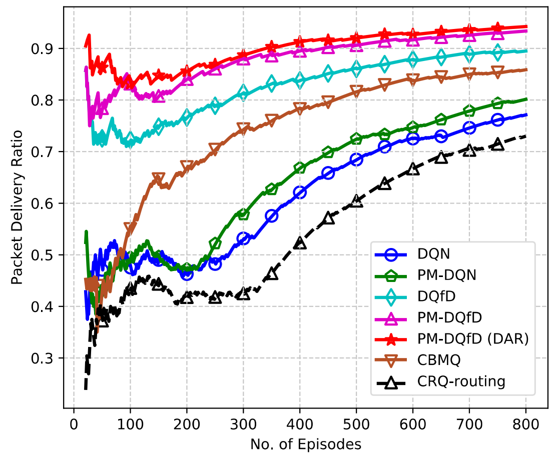
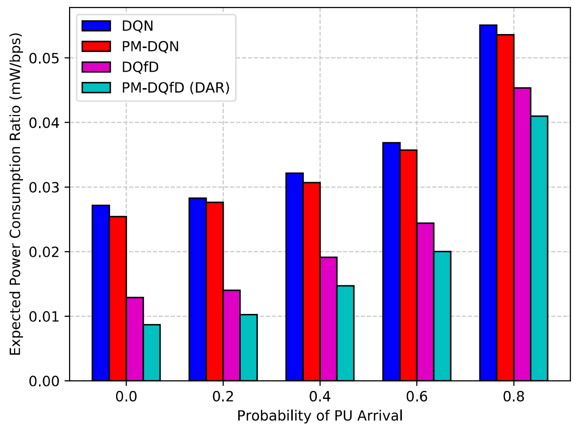
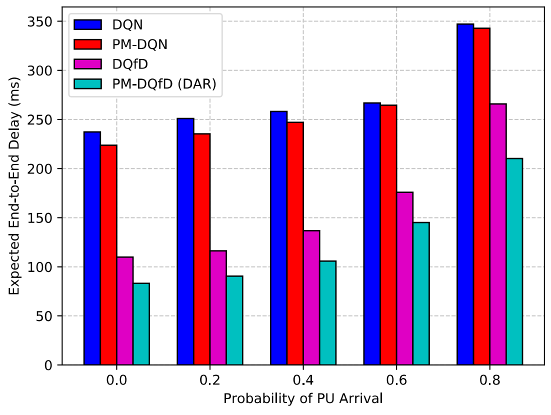


Figure 11. Packet delivery ratio versus number of episodes.

Figure 12a,b illustrate the trend of system PCR and routing latency as a function of PU arrival probability, respectively. In Figure 12a, the system PCR increases as PU arrival probability increases. This is because frequent arrival of PUs results in more data transmissions being interrupted. Increase in retransmission results in greater energy expenditure. For different PU arrival probabilities, PM-DQfD achieves the lowest system PCR, followed by DQfD. The PCR of PM-DQN is obviously higher than DQfD, and DQN has the greatest energy consumption. This demonstrates the advantage of learning from the expert demonstrations and memory structure optimization. Furthermore, it can be seen that the advantage of DQfD-based schemes decreases as PU arrival probability increases. This occurs mainly because the performance of the expert agent tends to deteriorate when the primary user returns frequently. Thus, even with the aid of the expert's demonstration data, the advantage to the apprentice node is still diminished. In addition, the change end-to-end delay with PU arrival probability given in Figure 12b is very similar to that in Figure 12a, and we will not repeat it here.



(**a**) Expected power consumption ratio



(**b**) Expected end-to-end delay

Figure 12. (**a**,**b**) Effectiveness properties versus probability of PU arrival.

5.3. Time Complexity Analysis

In this section, the time complexity of the cross-layer routing protocol is taken into consideration. The neural network has three fully connected layers in the simulation experiment. The number of neurons in the input and output layer is equal to the dimension of the state $n_{feature}$ and the size of action space n_{action}, respectively. The neuron quantities in the first and second layer are n_1 and n_2, respectively. The input layer is not included in the total number of layers in the neural network because it only transmits data into the network and does not participate in the calculation. The feed-forward calculation for one sample contains two matrix operations, which needs $n_1 \times n_2$ and $n_2 \times n_{action}$ times of computation [20]. Since the size of the action space n_{action} is a constant, the time complexity for one sample in the feed-forward calculation can be calculated as $o(n_1 \times n_2 + n_2 \times n_{action}) = o(n_1 \times n_2)$. The complexity of the back propagation is the same as that of the feed-forward calculation, such that the time complexity of training the neural network for one sample is $o(n_1 \times n_2)$. We assume that the algorithm converges after T episodes, and the average hops along one path are H (it can be seen that T is 800 and H approximates to 5 from the simulation results in Figure 10). Thus, the total number of samples in the simulation process is $T \times H$, and the time complexity of the apprenticeship learning-based scheme is $o(T \times H \times n_1 \times n_2)$. Since the values of both T and H are less than those in other algorithms, the proposed routing protocol has a lower time complexity than the self-learning schemes.

5.4. Discussion of Application Scenario

On the basis of experiment and evaluation, it is found that the apprenticeship learning-based cross-layer routing protocol has the following characteristics: Firstly, its learning speed is sufficiently high with the guidance of expert demonstration data. Secondly, it achieves high energy efficiency by reducing the trial-and-error cost and applying the DAR. Thirdly, the routing latency of the scheme is relatively low, and it accommodates the maximal latency requirement. Fourthly, it releases memory space periodically to store more data caches or high-precision data. Fifthly, by using DRL, the proposed algorithm is able to solve joint routing and resource management for large-scale networks. With all these advantages, the proposed scheme can be used in wireless sensor networks such as the Internet of Things (IoT) and the Wireless Environmental Monitoring Network (WEMN). In these applications, the number of sensor nodes is large, and every node has limited storage resources. Moreover, the network lifetime is limited, since the nodes are energy-constrained and the environment of their deployment area is complicated. In addition, the proposed scheme can also be applied in cognitive radio ad hoc networks such as Military Ad Hoc Networks (MAHN) and Personal Area Networks (PAN). These applications require the network to have fast self-organizing capability, and the routing protocol has to ensure that the end-to-end delay is within the tolerance and meets the application requirement. Furthermore, since most radios rely on batteries for power supply, energy efficiency should be taken into account. Consequently, the proposed scheme would satisfy all of the requirements for these application scenarios and produce better network performance.

6. Conclusions

Green cognitive radio technology has attracted increasing attention due to the energy efficiency requirements of beyond-5G wireless communication systems. It achieves reasonable tradeoffs between energy efficiency and spectrum efficiency in wireless networks. In this paper, we studied the key techniques in the green cognitive radio, and a PM-DQfD-based energy-efficient cross-layer routing scheme was presented. Unlike the general DRL method, PM-DQfD utilizes expert demonstrations to accelerate the learning procedure, which improves both of the energy efficiency and spectrum efficiency. Furthermore, it reduces storage occupancy and achieves better performance through memory structure optimization. This approach is especially suitable for the energy-constrained and delay-limited applications in green communications. Although energy efficiency is considered one of the most important performance metrics in this paper, SUs are battery-powered and energy-constrained. Once the power of the SU nodes starts to run out, the function of the whole network will be affected. Nowadays, energy harvesting technology is being studied and is becoming available that is able to extract energy from the surrounding environment. This can greatly improve the lifetime of wireless networks and reduce the difficulty of node deployment. In future work, we aim to apply energy harvesting to the joint routing and resource management in CRN.

Author Contributions: The individual contributions of authors are as follows. Y.D. developed the joint design scheme and wrote the paper. Y.X. directed the research and contributed to the refinement of the algorithm. The analysis and performance discussion were performed by L.X. The paper was subsequently revised by L.W. and F.Z.

Funding: This research was funded by the National University of Defense Technology Scientific Research Project (No. JS17-03-38).

Acknowledgments: The authors wish to thank the editor and anonymous referees for their helpful comments in improving the quality of this paper.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Ping, S.; Aijaz, A.; Holland, O.; Aghvami, A.H. SACRP: A spectrum aggregation-based cooperative routing protocol for cognitive radio ad-hoc networks. *IEEE Trans. Commun.* **2015**, *63*, 2015–2030. [CrossRef]
2. Hoan, T.N.K.; Koo, I. Multi-slot spectrum sensing schedule and transmitted energy allocation in harvested energy powered cognitive radio networks under secrecy constraints. *IEEE Sens. J.* **2017**, *17*, 2231–2240. [CrossRef]
3. Darsena, D.; Gelli, G.; Verde, F. An opportunistic spectrum access scheme for multicarrier cognitive sensor networks. *IEEE Sens. J.* **2017**, *17*, 2596–2606. [CrossRef]

4. Wu, Y.; Hu, F.; Kumar, S.; Zhu, Y.; Talari, A.; Rahnavard, N.; Matyjas, J.D. A learning-based qoe-driven spectrum handoff scheme for multimedia transmissions over cognitive radio networks. *IEEE J. Sel. Areas Commun.* **2014**, *32*, 2134–2148. [CrossRef]
5. Khan, U.A.; Lee, S.S. Multi-Layer Problems and Solutions in VANETs: A Review. *Electronics* **2019**, *8*, 204. [CrossRef]
6. Yang, Z.; Cheng, G.; Liu, W.; Yuan, W.; Cheng, W. Local coordination based routing and spectrum assignment in multi-hop cognitive radio networks. *Mob. Netw. Appl.* **2008**, *13*, 67–81. [CrossRef]
7. Cheng, G.; Liu, W.; Li, Y.; Cheng, W. Joint on-demand routing and spectrum assignment in cognitive radio networks. In Proceedings of the 2007 IEEE International Conference on Communications, Glasgow, UK, 24–28 June 2007; pp. 6499–6503.
8. Wang, J.; Yue, H.; Hai, L.; Fang, Y. Spectrum-aware anypath routing in multi-hop cognitive radio networks. *IEEE Trans. Mob. Comput.* **2017**, *16*, 1176–1187. [CrossRef]
9. Li, D.; Lin, Z.; Stoffers, M.; Gross, J. Spectrum Aware Virtual Coordinates Assignment and Routing in Multihop Cognitive Radio Network. In Proceedings of the 14th IFIP International Conference on Networking, Toulouse, France, 20–22 May 2015.
10. Zhang, L.; Cai, Z.; Li, P.; Wang, X. Exploiting spectrum availability and quality in routing for multi-hop cognitive radio networks. In Proceedings of the International Conference on Wireless Algorithms, Systems, and Applications, Bozeman, MT, USA, 8–10 August 2016; Springer: Cham, Germany, 2016; pp. 283–294.
11. Mansoor, N.; Islam, A.M.; Zareei, M.; Baharun, S.; Komaki, S. A novel on-demand routing protocol for cluster-based Cognitive Radio ad-hoc Network. In Proceedings of the TENCON 2016-2016 IEEE Region 10 Conference, Singapore, 22–25 November 2016.
12. Amini, R.M.; Dziong, Z. An economic framework for routing and channel allocation in cognitive wireless mesh networks. *IEEE Trans. Netw. Serv. Manag.* **2014**, *11*, 188–203. [CrossRef]
13. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G.; et al. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529. [CrossRef]
14. Sutton, R.S.; McAllester, D.A.; Singh, S.P.; Mansour, Y. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*; MIT Press: Cambridge, MA, USA, 2000; pp. 1057–1063.
15. Lillicrap, T.P.; Hunt, J.J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; Wierstra, D. Continuous control with deep reinforcement learning. *arXiv* arXiv:1509.02971, 2015.
16. Koushik, A.M.; Hu, F.; Kumar, S. Intelligent spectrum management based on transfer actor-critic learning for rateless transmissions in cognitive radio networks. *IEEE Trans. Mob. Comput.* **2017**, *17*, 1204–1215.
17. Sendra, S.; Rego, A.; Lloret, J.; Jimenez, J.M.; Romero, O. Including artificial intelligence in a routing protocol using software defined networks. In Proceedings of the 2017 IEEE International Conference on Communications Workshops (ICC Workshops), Paris, France, 21–25 May 2017; pp. 670–674.
18. Yu, C.; Lan, J.; Guo, Z.; Hu, Y. DROM: Optimizing the routing in software-defined networks with deep reinforcement learning. *IEEE Access* **2018**, *6*, 64533–64539. [CrossRef]
19. Mustapha, I.; Ali, B.; Rasid, M.; Sali, A.; Mohamad, H. An energy-efficient spectrum-aware reinforcement learning-based clustering algorithm for cognitive radio sensor networks. *Sensors* **2015**, *15*, 19783–19818. [CrossRef] [PubMed]
20. Du, Y.; Zhang, F.; Xue, L. A Kind of Joint Routing and Resource Allocation Scheme Based on Prioritized Memories-Deep Q Network for Cognitive Radio Ad Hoc Networks. *Sensors* **2018**, *18*, 2119. [CrossRef] [PubMed]
21. Karmokar, A.; Naeem, M.; Anpalagan, A.; Jaseemuddin, M. Energy-efficient power allocation using probabilistic interference model for OFDM-based green cognitive radio networks. *Energies* **2014**, *7*, 2535–2557. [CrossRef]
22. Singh, K.; Moh, S. An energy-efficient and robust multipath routing protocol for cognitive radio ad hoc networks. *Sensors* **2017**, *17*, 2027. [CrossRef] [PubMed]
23. Du, Y.; Chen, C.; Ma, P.; Xue, L. A Cross-Layer Routing Protocol Based on Quasi-Cooperative Multi-Agent Learning for Multi-Hop Cognitive Radio Networks. *Sensors* **2019**, *19*, 151. [CrossRef]

24. Alom, M.Z.; Taha, T.M.; Yakopcic, C.; Westberg, S.; Sidike, P.; Nasrin, M.S.; Hasan, M.; Van Essen, B.C.; Awwal, A.A.S.; Asari, V.K. A State-of-the-Art Survey on Deep Learning Theory and Architectures. *Electronics* **2019**, *8*, 292. [CrossRef]
25. Wang, W.; Kwasinski, A.; Niyato, D.; Han, Z. A survey on applications of model-free strategy learning in cognitive wireless networks. *IEEE Commun. Surv. Tutor.* **2016**, *18*, 1717–1757. [CrossRef]
26. Wu, Y.; Hu, F.; Kumar, S.; Matyjas, J.D.; Sun, Q.; Zhu, Y. Apprenticeship learning based spectrum decision in multi-channel wireless mesh networks with multi-beam antennas. *IEEE Trans. Mob. Comput.* **2017**, *16*, 314–325. [CrossRef]
27. Wu, Y.; Hu, F.; Zhu, Y.; Kumar, S. Optimal spectrum handoff control for CRN based on hybrid priority queuing and multi-teacher apprentice learning. *IEEE Trans. Veh. Technol.* **2017**, *66*, 2630–2642. [CrossRef]
28. Hester, T.; Vecerik, M.; Pietquin, O.; Lanctot, M.; Schaul, T.; Piot, B.; Dulac-Arnold, G. Deep Q-learning from demonstrations. In Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, LA, USA, 2–7 February 2018; pp. 3223–3231.
29. Hester, T.; Vecerik, M.; Pietquin, O.; Lanctot, M.; Schaul, T.; Piot, B.; Sendonaris, A.; Dulac-Arnold, G.; Osband, I.; Agapiou, J.; et al. Learning from demonstrations for real world reinforcement learning. *arXiv* **2017**, arXiv:1704.03732.
30. Andre, D.; Friedman, N.; Parr, R. Generalized prioritized sweeping. In *Advances in Neural Information Processing Systems*; University of California: Berkeley, CA, USA, 1998; pp. 1001–1007.
31. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; Riedmiller, M. Playing Atari with Deep Reinforcement Learning. Computer Science. *arXiv* arXiv:1312.5602v1, 2013.
32. Pourpeighambar, B.; Dehghan, M.; Sabaei, M. Non-cooperative reinforcement learning based routing in cognitive radio networks. *Comput. Commun.* **2017**, *106*, 11–23. [CrossRef]
33. Al-Rawi, H.A.; Yau, K.L.A.; Mohamad, H.; Ramli, N.; Hashim, W. A reinforcement learning-based routing scheme for cognitive radio ad hoc networks. In Proceedings of the 2014 7th IFIP Wireless and Mobile Networking Conference (WMNC), Vilamoura, Portugal, 20–22 May 2014; pp. 1–8.
34. Chen, X.; Zhao, Z.; Zhang, H. Stochastic Power Adaptation with Multiagent Reinforcement Learning for Cognitive Wireless Mesh Networks. *IEEE Trans. Mob. Comput.* **2013**, *12*, 2155–2166. [CrossRef]