Partially Observable Double DQN Based IoT Scheduling for Energy Harvesting

Dongji Li, Shaoyi Xu, and Jianan Zhao
School of Electronics and Information Engineering, Beijing Jiaotong University, Beijing, China
E-mail:{dongjili, shyxu, jiananzhao}@bjtu.edu.cn

***Abstract*—Energy harvesting (EH) is a promising technique to extend the lifetime of internet of things (IoT) networks. In real cases however, its costly and impractical that each node send its data to base station each slot, thus we decided to sample the data of some nodes and reconstructed the state of all nodes through the belief state in order to reduce energy cost. In this paper, we consider a dynamic multiple channel access problem in a single cell. There are multiple nodes and each of them is equipped with an EH device and a rechargeable battery. Particularly, we consider that a BS only observes the power information of scheduled nodes. Thus the whole problem is modeled as a partially observable Markov decision process (POMDP). We firstly convert the observation of partial nodes to the belief state of all nodes is proposed to predict the scheduling policy of the next time slot. And then we propose a deep reinforcement learning algorithm called Double deep Q network (Double DQN). Simulation results has clearly indicated that the performance of our proposed Double DQN outperforms that of other reinforcement learning (RL) algorithms.**

***Index Terms*—EH communication, PODMP, Deep Reinforcement learning, Double DQN**

I. Introduction

As a key driver for 5G, internet of things (IoT) is gaining popularity. Ericsson predicts that there will be around 28 billion connected devices by 2021 [1]. Therefore, the lifetime of devices becomes the major bottleneck in the development of IoT. However, it is difficult and impractical to replace or charge the battery of devices in some places, especially in some hard-to-reach places and dead zones. Energy harvesting (EH) enables devices to acquire energy continually from nature or the man-made phenomena. This is a promising technology to extend networks lifetime significantly and efficiently reduce the green housing gas emissions [2]. More and more researchers are attempting to fulfill the long-term or self-sustainable operations for IoT systems using EH technique. Due to the random nature of energy, the lack of accurate predictions of the energy harvesting process is seen as a huge problem. The allocation of energy used to transmit data and stored in battery must be considered by nodes to extend its lifetime and maximize the throughput of the whole system simultaneously. Meanwhile, base station (BS) must consider that which nodes should be scheduled to transmit data. Hence, it is vital to optimize the scheduling policy of IoT networks with the dynamics of harvested energy and multiple channels.

In recent years, the research about EH wireless communication systems has attracted considerable interests, especially in scheduling policies that exploit and explore the scarce spectrum resource and harvested energy in the most efficient manner. Actually, the EH communication system can be modeled as a Markov decision process (MDP) [3], or as a partially observable Markov decision process (POMDP) [4], both dynamic programming (DP) and reinforcement learning (RL) can be used to address this problem [5]. A learning theoretic approach is proposed in [6], which does not consider any a priori information on the Markov processes governing the communication system, to maximize the expected total data transmitted during the transmitters lifetime. Besides [6] also studies online and offline optimization problems additionally under the same assumption. [7] proposes a novel energy management algorithm based on reinforcement learning named RLMan, which automatically adopts an optimal energy management policy in the time-varying environment. Authors in [8] constructed a two-level RL network to maximize the sum rate and minimize the prediction loss simultaneously. Furthermore they applied a long short-term memory (LSTM) DQN based approach to design the uplink access control policy and predict the capacity of battery. However, all of the aforementioned papers only take modeling problem as MDP into consideration. Unfortunately, it is obviously impossible that a BS obtains all information about communication system and EH process in IoT networks. In order to deal with this problem, authors in [9] have modeled the whole problem as a POMDP to schedule EH nodes to access the multi-channel communication system. They show the optimality of myopic policy (MP) for large low-power wireless sensor networks under certain assumptions. [10] studies optimality of MP and whittle indexability for energy harvesting sensors networks, and shows the performance of MP and whittle indexability in some special cases. Besides, as one of the long-standing challenges of reinforcement learning, high-dimensional input incurs insufficient capacity and computing power. Then, the POMDP is PSPACE-hard and the best known solution for finding the exact solution requires an exponential computation complexity [11].

Previous published studies are limited to some special cases. So far, few studies have investigated in solving POMDP in general cases. Consequently, we propose a deep reinforcement learning named double deep Q network (Double DQN) to deal with this problem in general cases. we study the case that the each node is equipped with EH device and a rechargeable bat-

978-1-7281-2373-8/19/$31.00 ©2019 IEEE

tery with limited capacity in an IoT network. In this wireless communication system, BS only observes power information about partial nodes, thus the MDP is not suitable to this problem any more. Therefore this problem is modeled as a POMDP and solved numerically and solved by Double DQN. Especially, Double DQN improves the performance through reducing the overestimate caused by Q learning. The main contributions of this paper are summarized as follows:

- To resolve the scheduling problem caused by partially observable information in EH wireless communication system , we propose a deep reinforcement learning algorithm.
- In order to obtain the better solution, we propose Double DQN to reduce overestimate caused by Q-learning.
- To the best of our knowledge, we are the first to propose Double DQN for the EH wireless communication system with observable information of partial nodes.

The remainder of this paper is organized as follows. The system model and problem formulation are presented in Section II. Section III is dedicated to study dynamic multiple channel access using Double DQN. Section IV presents the performance of Double DQN and analyzes our simulation results. Finally, the paper is concluded in Section V.

II. System Model and Problem Formulation

A. Symtem Model

As depicted in Fig.1, we consider N nodes and each node equipped with an EH device and a rechargeable battery in a single cell, and the maximal capacity of nodes' battery is C. There are a BS and K orthogonal channels in this cell, where BS is responsible for the uplink scheduling operations. In each time slot (TS), an orthogonal channel can be occupied by only one node, and the channle gain is constant during each TS. We assume the energy arrive process is a memoryless Poisson process with the rate of λ. Let $\mathbf{e(t)} = [e_1(t), \ldots, e_i(t), \ldots, e_N(t)]$ represent the vector of energy harvested by all nodes in TS t, where $e_i(t)$ denotes harvested energy of node i in TS t. The EH process is modeled as a two-state Markov process, as shown in Fig.2. We use p_{ij} to represent the state transition probability from state i to j, and assume $p_{11} > p_{01}$, because the EH process is positively correlated in time [9], i.e. nodes are willing to maintain the same state as the previous TS. Harvested state and non-harvested state are denoted by 0 and 1 respectively. The successful transmission threshold (energy consumed for successful transmission) d is fixed.

At the beginning of each TS, BS produces the scheduling policy $u(t)$ with the probability ϵ except the first TS (in the first TS, BS schudules randomly), or schedules randomly with the probability $1-\epsilon$. Afterward, BS broadcasts the scheduling policy $u(t)$ to all nodes, then receives the information about current power $h(t)$ of scheduled nodes. The scheduled nodes send back the information of their residual power $h'(t)$ to BS again after attempting to transmit data. It's worth noting that a node could transmit data only if it satisfies two conditions:

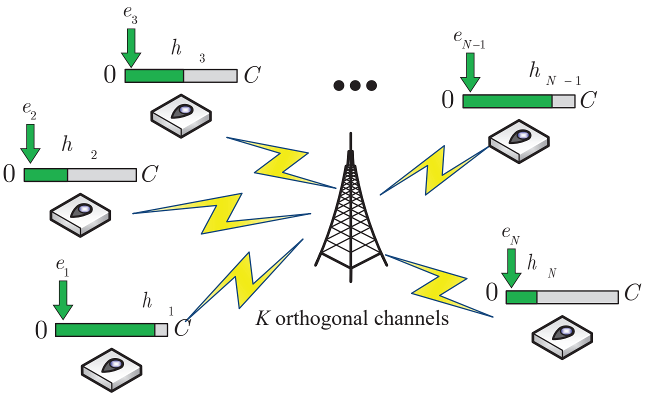


Fig. 1: Wireless sensor network with N EH nodes and K orthogonal channels.

1) its power is more than d; 2) it is scheduled in TS t. Moreover, there is a historical scheduling table in BS to store the historical information about scheduled nodes for planning the schedule of the next TS.

B. Problem Formulation

Generally, the most of problems can be modeled as a MDP when BS can observe the information about total nodes' battery. And Q-learning can be used to deal with these problems without the statistics about EH process, i.e. Q-learning is a model-free reinforcement learning algorithm. Let $S(t) = [h_1(t), \ldots, h_i(t), \ldots, h_N(t)]$ denote the state in TS t, where $h_i(t)$ is the power information about the scheduled node i. We replace action with the policy $u(t)$ in TS t, therefore the dimension of the action space is equal to the number of combinations of K from N, namely C_N^K. The immediate reward r is given as 채널 노드

$$r(t) = \sum_{i \in u(t)} \left| h_t(i) - h_i^{'}(t) \right|. \tag{1}$$

Our goal is maximizing total expected discounted accumulated reward during lifetime T TSs, and it can be defined as

$$\begin{aligned} &\max_{\{u(t)\}_{t=1}^{T}} \mathbb{E}\left[\sum_{t=1}^{T} \beta^{t-1} r(t)\right] \\ &s.t. \quad h_i(t+1) = \min\left\{h_i^{'}(t) + e_i(t), C\right\} \cdot 1\left(i \in \mathcal{U}(t)\right) \\ &\qquad + \min\left\{h_i(t) + e_i(t), C\right\} \cdot 1\left(i \notin \mathcal{U}(t)\right), \end{aligned} \tag{2}$$

where $0 \leq \beta \leq 1$ is a discount factor, and $\mathbf{1}\left(\cdot\right)$ is an indicator function, i.e. $\mathbf{1}$(ture)=1 and $\mathbf{1}$(false)=0. The constraint of (2) prevents the power of node i from overflowing.

Then we rewrite (2) using value action function in order to get the optimal policy,

$$Q_\pi(s_i, u) = \mathbb{E}_\pi\left[\sum_{t=1}^{T} \beta^{t-1} r(t) | s_0 = s_i, a_0 = u\right]. \tag{3}$$

We obtain the optimal policy π^* through maximizing (3),

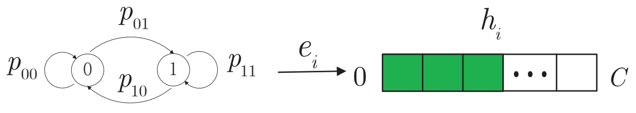


Fig. 2: Two-state Markov process.

$$\pi^* = \underset{u\in\mathcal{U}}{\operatorname{argmax}}\, Q_\pi(s_i,u). \tag{4}$$

With the property of value action function, (3) can be evaluated in a recursive manner using Bellman equations, wherefore (3) is given as

$$\begin{aligned} Q_{\pi^*}(s_i,u) &= \mathbb{E}\left[R(t) + \beta \max_{u'} Q_{\pi^*}(s_i^{'},u^{'})\right] \\ &= \sum_{s_i^{'}} p(s_i^{'}|s_i,u)\left[g(s_i^{'},u,s_i) + \beta \max_{u'} Q_{\pi^*}(s_i^{'},u^{'})\right], \end{aligned} \tag{5}$$

where $p(s_i^{'}|s_i,u)$ is the transition probability that the state of node i transits from s_i to $s_i^{'}$ under action u. $g(s_i^{'},u,s_i)$ is the expected reward of the next reward given the current state s_i and action u, together with the next state $s_i^{'}$.

In practice, it is difficult to obtain the exact transition probability of MDP. Fortunately, we can found the optimal strategy based on the available information $(s_i,u,r,s_i^{'})$ in a recursive manner through Monte-Carlo (MC) methods. The basic idea of MC methods is to acquire certain episodes under some policies, then sample and learn directly from episodes, in order to use empirical average return instead of expected return [12]. Thus (5) is rewritten as

$$Q_\pi(s_i,u) = Q_\pi + \gamma(G_\pi - Q_\pi(s_i,u)), \tag{6}$$

where G_π is the total expected reward of episodes under policy π and $0 \le \gamma \le 1$ denotes learn rate. Nevertheless, MC methods must wait until the end of the complete episode to know the return. In other words, the BS has to wait until the whole episode is calculated completely. There is a lot of time to waste by using MC methods. Fortunately, Temporal differences (TD) methods can learn from the continuous environments without the final outcome [13]. For this reason, we decide to adopt the Q-learning with TD methods as follows:

$$\begin{aligned} Q_\pi(s_i,u) = {} & Q_\pi(s_i,u) + \\ & \gamma\left(r + \beta \max_{u'} Q_{\pi^*}(s_i^{'},u^{'}) - Q_\pi(s_i,u)\right), \end{aligned} \tag{7}$$

where r means the immediate reward.

C. Partially Observable Markov Decision Process

In our paper, due to the fact that BS is impossible to learn battery information of all nodes, we model the problem as a POMDP and repalce the state $s(t)$ with the observation $o(t) = [\ldots, h_i(t), \ldots], i \in u(t)$ to express the fact that BS only observes the power information of partial nodes, i.e. BS only learns the power information of scheduled nodes. However it is hard to obtain an optimal policy with the partial information, so we need to extend the observation of scheduled nodes to the state of all nodes. For this reason, we introduce the belief state $b(t) = [b_1^1(t), \ldots, b_i^x(t), \ldots, b_N^C(t)]$, where $b_i^x(t)$ demonstrates the probability that the battery capacity of node i is x in TS t. The belief state of node i in TS t is

$$b_i^x(t) = \begin{cases} \frac{b_i^x(t-1)\cdot g_i(t)}{\sum_{j=1}^M b_i^x(t-j)} & i \in u(t), \quad o_i t = x \\ \frac{b_i^x(t-1)\cdot h_i(t)}{\sum_{j=1}^M b_i^x(t-j)} & i \in u(t), \quad o_i t \neq x \\ \frac{b_i^x(t-1)\cdot \frac{1}{C}}{\sum_{j=1}^M b_i^x(t-j)} & i \notin u(t), \end{cases} \tag{8}$$

where M is the length of the historical scheduling table. What should be noticed that the capacity of nodes is not less than one unit. In (8), the first two lines reflect the fact that the belief state of scheduled node becomes clear, i.e. it is easy to determine which state the scheduled node is in. We use $g_i(t) = \frac{o_i t + 0.1}{C}$ and $h_i(t) = \frac{1-g_i(t)}{C-1}$ to represent the changing trend of the scheduled nodes' belief state. For example, although both $b_i^x(t)$ and $b_i^y(t)$ are the belief states of the scheduled node i, the $b_i^x(t)$ is greater than $b_i^y(t)$ when the observation of scheduled node i is equal to x rather than y. It is difficult to judge which state the non-scheduled node is in, because the belief states of non-scheduled nodes is updated by multiplying by ? the same being? factor $\frac{1}{C}$. In order to prevent the belief state from bing equal to zero, a small number 0.1 is added to the numerator of the first equation. And we assume the initial belief state b(1) is

$$b(1) = [b_1^1, \ldots, b_i^x, \ldots, b_N^C] = [\underbrace{\frac{1}{C\times N}, \ldots, \frac{1}{C\times N}}_{C\times N}]. \tag{9}$$

$S_i \to b_i$

Then (7) is rewritten as

$$\begin{aligned} & Q_\pi(b_i,u|b(1)) = Q_\pi(b_i,u|b(1)) = + \\ & \alpha\left(r + \beta \max_{u'} Q_{\pi^*}(b_i^{'},u^{'}|b(1)) - Q_\pi(b_i,u|b(1))\right). \end{aligned} \tag{10}$$

III. Double DQN and Multiple Access Control

A. Double Deep Q Network

As one of the most popular deep reinforcement learning, DQN has been used widely since 2015 [14]. The emergence of DQN makes it possible to address complex control problems, especially for the high-dimension cases. DQN is two multi-layered neural networks, including an evaluate net with the parameter θ and a target net with the parameter $\theta^{'}$. The output of target net is used as a label to train the evaluate net. Loss function is a key part of DQN, which builds a brige between the evaluate net and the target net. Loss function of DQN is defined as

$$\begin{aligned} Loss^{DQN} = & (r + \gamma Q(s, \underset{a}{\operatorname{argmax}}\, Q(s^{'},\cdot,\theta^{'}),\theta^{'}) \\ & - Q(s,a,\theta)), \end{aligned} \tag{11}$$

where $(s,a,r,s^{'})$ is experience picked from *experience reply* pool randomly to break correlations among data and make training stable and convergent.

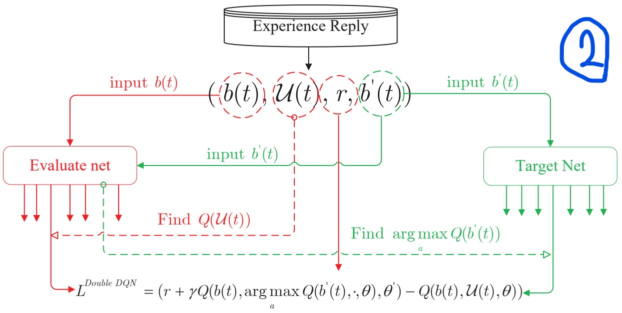

Experience Reply
input $b(t)$
$(b(t), \mathcal{U}(t), r, b^{'}(t))$
input $b^{'}(t)$
input $b^{'}(t)$
Evaluate net
Target Net
Find $Q(\mathcal{U}(t))$
Find $\arg\max_a Q(b^{'}(t))$
$L^{Double\ DQN} = (r + \gamma Q(b(t), \arg\max_a Q(b^{'}(t), \cdot, \theta), \theta^{'}) - Q(b(t), \mathcal{U}(t), \theta))$

Fig. 3: Simplified architecture of Double DQNs loss function.

Both Q-learning algorithm and DQN algorithm are known to overestimate the action value under certain conditions owing to insufficiently flexible function approximation and noise [15]. When overestimate does not occur uniformly, the quality of the resulting policy could be affected negatively [16]. Hence, in 2015, Google DeepMind proposed a new algorithm called Double DQN [17] based on the Double Q-learning algorithm proposed by Hasselt in 2010 [18]. Google DeepMind modifies the loss function of Double DQN to reduce overestimate and the Loss function of Double DQN is given as

$$\begin{aligned} Loss^{DoubleDQN} =& (r + \gamma Q(s, \underset{a}{\text{argmax}}\, Q(s^{'}, \cdot, \theta), \theta^{'}) \\ & - Q(s, a, \theta)). \end{aligned} \tag{12}$$

Fig.3 illuminates the simplified architecture of Double DQN loss function. Through adopting mini-batch gradient descent (MBGD) to update the evaluate net, we get the following gradient:

$$\begin{aligned} \bigtriangledown_{\theta} Loss^{DoubleDQN} =& (r + \gamma Q(s, \underset{a}{\text{argmax}}\, Q(s^{'}, \cdot, \theta), \theta^{'}) \\ & - Q(s, a, \theta)) \bigtriangledown_{\theta} Q(s, a, \theta), \end{aligned} \tag{13}$$

where $\bigtriangledown_{\theta} f(\cdot)$ denotes the gradient vector of $f(\cdot)$ with respect to θ.

B. Dynamic multiple channel access

In our paper, Double DQN architecture is deployed at BS in order to reduce the energy consumption of nodes. The process of dynamic multiple channel access is divided into two phases including to predict scheduling policy and to train the Double DQN.

1 2

1

To predict the scheduling policy, BS chooses the scheduling policy that has the highest output value from the evaluate net as the optimal policy *u(t)* with the probability ϵ or schdules randomly with the probability $1 - \epsilon$. After that, BS observes the current power of scheduled node i as $o_i(t)$ and the residual power as $o_i^{'}(t)$ in TS t and calculates the immediate reward r using (1). Then $o_i(t)$ and $o_i^{'}(t)$ are converted to $b_i(t)$ and $b_i^{'}(t)$ by using (8).

2

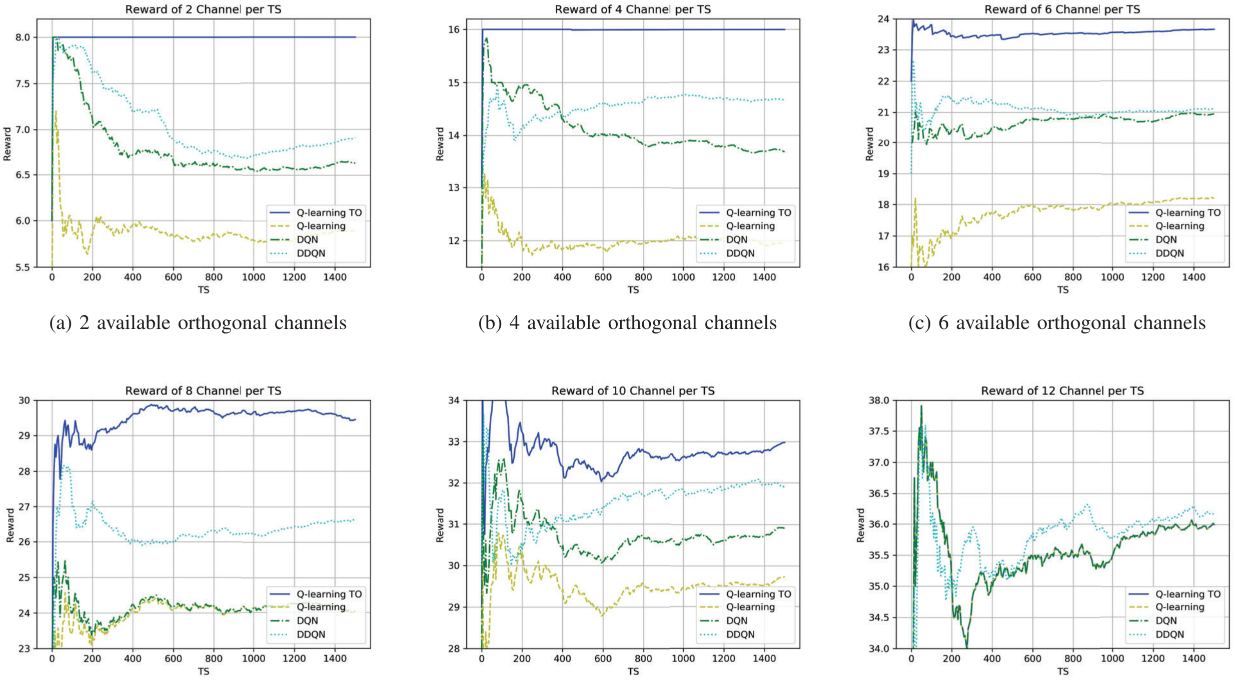
In the train phase, $(b(t), u(t), r, b^{'}(t))$ is constructed as experience. Before this experience is stored into experience reply pool, Double DQN has to check if it exists in experience reply pool firstly. If the experience reply pool does not have this experience, it will be stored. New experience covers the oldest experience circularly in order to prevent experience reply pool from overflowing. Double DQN picks a mini-batch number of experience from experience reply pool randomly to train the evaluate net when the number of experience exceeds the quantity of a mini-batch. Double DQN has been trained using (7) for m times every batch. The weights of the target net $\theta^{'}$ is copied by that of evaluate net θ every L TSs. The overall scheduling process is summarized in Algorithm 1.

Algorithm 1 Dynamic multiple channel access based on partially observable Double DQN.

```
Initialize the historical scheduling table and experience reply pool;
Initialize the weights of evaluate net and that of target net;
for $t = 1$ to $T$ do
Predict scheduling policy phase:
    if this is first TS then
        BS selects scheduling policy randomly and broadcasts to all nodes;
action
    else
        if randomly generate a float number between 0 and $1 \leq \epsilon$ then
ε           BS selects the optimal scheduling policy from evaluate net and broadcasts to all nodes;
        else
            BS selects scheduling policy randomly and broadcasts to all nodes;
        end if
    end if
    Scheduled nodes send the information of their power as $o(t)$ and try to transmit data, then send their residual power information again as $o^{'}(t)$;
    All nodes harvest energy and store it for the next TS;
Double DQN train phase:
    BS converts $o(t)$ and $o^{'}(t)$ to $b(t)$ and $b^{'}(t)$ according to (8), and calculates immediate reward $r$ using (1);
    BS constructs the experience $(b(t), u(t), r, b^{'}(t))$ and stores it in experience reply pool;
    Double DQN randomly picks a mini-batch of experience from experience reply pool to train the evaluate net;
    $\theta^{'}$ is copied by weights of the evaluate net $\theta$ every $L$ TSs;
end for
```

IV. Simulation and Analysis

In this section, we provide the simulation results of our proposed partially observable Double DQN algorithm. Moreover, we compare the performance of our algorithm with that of DQN with partial observation, Q-learning with partial observa-



(a) 2 available orthogonal channels

(b) 4 available orthogonal channels

(c) 6 available orthogonal channels

(d) 8 available orthogonal channels

(e) 10 available orthogonal channels

(f) 12 available orthogonal channels

Fig. 4: Reward per TS of different number of available orthogonal channels

tion and Q-learning with total observation. Q-learning with total observation is expressed as *Q-learning TO*, and Q-learning with partial observation, DQN with partial observation and Double DQN with partial observation are represented as *Q-learning*, *DQN* and *Double DQN*, respectively. Q-learning with total observation can be seen as the optimal benchmark of our simulation.

A. System Parameter and Double DQN Architecture

In our simulations, we consider multiple orthogonal channels which ranging from 2 to 12, and the number of nodes is 12 namely N=12 in this cell. Nodes have the identical initial battery capacity and maximal battery capacity of 0 and 10 units, respectively. The parameter λ of poison process is 4 and the transition probabilities of the EH processes are $p_{11} = p_{00} = 0.9$. We measure the performance of the scheduling policies as the average reward per TS over a time horizon of $T = 1500$.

We implement Double DQN algorithm in *Keras*, which is a high-level neural network API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano. The architecture of Double DQN is initialized by two totally same neural networks. Each network is constructed by three fully connected layers deployed sequentially. The number of neurons in the hidden layer is half of the sum that the number of input layer neurons plus that of the output layer namely, $\frac{(C \cdot N)+C_N^K}{2}$. The activation function of each neuron is Rectified Linear Unit (ReLU), which is equal to itself when it is greater than zero, otherwise equals zero. The detailed hyperparameters of Double DQN are shown in Table I.

TABLE I: HYPERPARAMETERS OF DOUBLE DQN

Parameter	*Value*
Learn rate γ	0.001
Reward decay β	0.9
ϵ*-greedy factor*	0.9
mini-batch	32
Experience reply size	500
Optimizer	Adam
Activation Function	ReLu

B. Numerical Results

We firstly investigate the performance with respect to different number of available channels and show the results for K = 2, 4, 6, 8, 10 and 12 available orthogonal channels. Firstly, it is obvious that the reward per TS increases as the number of available orthogonal channels increases and all alrigothms converge after around 500 TSs. Secondly, *Q-learning TO* has the best performance in Fig.3 except (f) subplot. As a result it is regarded as the benchmark in our paper. Thridly, *Double DQN* significantly outperforms *Q-learning* and *DQN*,especially when it has more information about the power of scheduled nodes.

Finally, we want to emphasize that Double DQN is better than other algorithms due to the fact that it reduces the overestimate when all algorithm can get the same observation

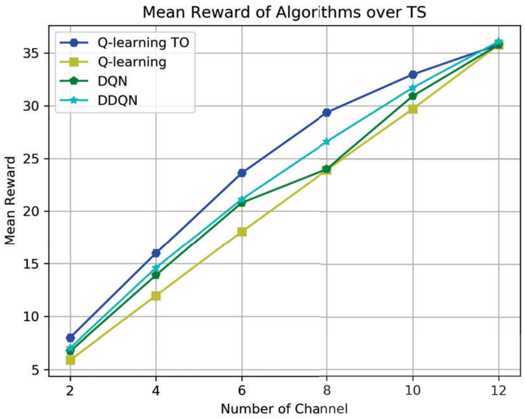


Fig. 5: Mean reward for the different number of available orthogonal channels.

of total nodes' power. Fig.5 illustrates the average reward for the different number of available orthogonal channels. After that, we observe the performance with respect to different learn rate. As we all know, learn rate is one of the most important parameters of a neural network to tune for training deep neural networks. Unfortunately, it is still an open question that how to choose the optimal learn rate and how to change it during training. Hence, we measure the performance of different learn rates $(lr = 0.001, 0.01, 0.05, 0.1 \text{ and } 0.2)$. For the result of our experiment, as decrease of learn rate, the performance of experiment is improved. Furthermore, it is very apparent that the best performance is obtained for $lr = 0.001$ in Fig.6, so we set the learn rate in our experiment to be 0.001.

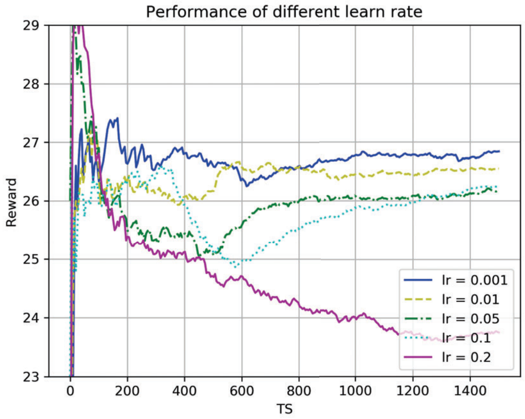


Fig. 6: Mean reward over TS of different learn rate.

V. Conclusion

In this paper, we propose a partially observable Double DQN to obtain the optimal scheduling policy of dynamic multiple channel access for EH wireless communication system. The whole problem has been modeled as a POMDP and designed to obtain the total expected discounted accumulated reward during lifetime. Our proposed Double DQN can provide a near-optimal solution with the information of only scheduled nodes' power. Particularly, when all algorithms can observe the complete power information of total nodes, our proposed Double DQN even outperforms other RL algorithms.

Acknowledgment

This work is supported in part by the National Natural Science Foundation of under grant 61571038 and the Fundamental Research Funds for the Central Universities with Grant 2016JBZ003.

References

[1] Ericsson, “White Paper: Cellular networksfor massive IoT,” January, 2016.

[2] S. Ulukus, A. Yener, and E. Erkip, et al, “Energy Harvesting Wireless Communications: A Review of Recent Advances, *IEEE Journal on Selected Areas in Communications,* vol. 33, pp. 360-381, March 2015.

[3] Z. Wang, A. Tajer, and X. Wang, “Communication of energy harvesting tags, *IEEE Transactions on Communications,* vol. 60, no. 4, pp. 1159-1166, Apr. 2012.

[4] A. Aprem, C. Murthy, and N. Mehta, “Transmit power control policies for energy harvesting sensors with retransmissions, *IEEE Journal of Selected Topics in Signal Processing,* vol. 7, no. 5, pp. 895-906, October. 2013.

[5] D. P. Bertsekas, “Dynamic Programming and Optimal Control, *Athenea Scientific,* 2005.

[6] P. Blasco, D. Gunduz, and M. Dohler, “A Learning Theoretic Approach to Energy Harvesting Communication System Optimization, *IEEE Transactions on Wireless Communications,* vol. 12, no. 4, pp. 1872-1882, April 2013.

[7] F. Ait Aoudia, M. Gautier, and O. Berder, “RLMan: An Energy Manager Based on Reinforcement Learning for Energy Harvesting Wireless Sensor Networks, *IEEE Transactions on Green Communications and Networking,* vol. 2, no. 2, pp. 408-417, June 2018.

[8] M. Chu, H. Li, X. Liao, and S. Cui, “Reinforcement Learning based Multi-Access Control and Battery Prediction with Energy Harvesting in IoT Systems, *IEEE Internet of Things Journal,* 2018.

[9] P. Blasco and D. Gndz, “Multi-Access Communications With Energy Harvesting: A Multi-Armed Bandit Model and the Optimality of the Myopic Policy, *IEEE Journal on Selected Areas in Communications,* vol. 33, no. 3, pp. 585-597, March 2015.

[10] F. Iannello, O. Simeone, and U. Spagnolini, “Optimality of myopic scheduling and whittle indexability for energy harvesting sensors, *2012 46th Annual Conference on Information Sciences and Systems (CISS),* Princeton, NJ, 2012, pp. 1-6.

[11] Papadimitriou, Christos H. and Tsitsiklis, John N, “The Complexity of Markov Decision Processes, *Mathematics of Operations Research,* vol. 12, pp. 441–450, 1987.

[12] R. Sutton, A. Barto, T. Dietterich, “Reinforcement Learning: an introduction, *MIT Press,* 1998.

[13] R. Sutton, Richard S, “Learning to predict by the methods of temporal differences, *Machine learning 3.1,* pp. 9-44, 1988.

[14] Mnih, Volodymyr, et al, “Human-level control through deep reinforcement learning, *Nature* pp, 518-529, 2015.

[15] Thrun, Sebastian, and Anton Schwartz, “Issues in using function approximation for reinforcement learning, *Proceedings of the 1993 Connectionist Models Summer School Hillsdale,* NJ, Lawrence Erlbaum, 1993.

[16] Kaelbling, Leslie Pack, Michael L. Littman, and Andrew W. Moore, “Reinforcement learning: A survey, *Journal of artificial intelligence research,* pp. 237-285, April 1996.

[17] Van Hasselt, Hado, Arthur Guez, and David Silver, “Deep Reinforcement Learning with Double Q-Learning, *Association for the Advance of Artificial Intelligence*, vol. 2, 2016.

[18] Hasselt, Hado V, "Double Q-learning," *Advances in Neural Information Processing Systems,* 2010.