Age of Information Minimization for Wireless Ad Hoc Networks: A Deep Reinforcement Learning Approach

Shiyang Leng and Aylin Yener

Wireless Communications and Networking Laboratory (WCAN)
School of Electrical Engineering and Computer Science
The Pennsylvania State University, University Park, PA 16802.
sfl5154@psu.edu *yener@engr.psu.edu*

***Abstract*—Age of information (AoI) has been recently considered as a performance metric to measure the freshness of information for time-critical wireless communications application. In this paper, we consider AoI minimization in a wireless ad hoc network, where nodes exchange status updates with one another over shared spectrum. The network needs to be formed in a dynamic fashion in the sense that each node either broadcasts or receives updates in a slot and attempts to keep the updates in both directions fresh. We aim to minimize the average AoI of each node by a joint broadcast scheduling and power control policy. Each node decides its transmitting/receiving mode and the transmission power based on its local observation of the system state. We formulate a Markov game and develop a multi-agent deep reinforcement learning algorithm based on deep recurrent Q-network. The simulation results show that the proposed approach outperforms the baselines significantly.**

I. Introduction

Age of information (AoI) has been recently introduced as a suitable metric to measure the freshness of information in wireless communication networks where timely information update is critical, for instance, beacon message exchange in vehicular networks [1]. AoI is defined as the time elapsed since the generation of the latest successfully received information. In [1], AoI of a point-to-point communication system is characterized and analyzed from a queuing theoretic perspective. It has been shown in [1] that AoI minimization offers different insights from delay minimization. Minimum AoI design in various systems since then has received extensive attention [2]–[6].

In this paper, we investigate AoI in a wireless ad hoc network. Although wireless ad hoc networks have been studied extensively over the decades, AoI minimization in wireless ad hoc networks is a new research direction critical in information transfer among peer-agents, for example, in the context of Internet of Things (IoT). Major application domains in this paradigm include wireless sensor networks and vehicular ad hoc networks. The general service requirement of such networks is to periodically update information among peer nodes on a short timescale in order to convey the states of the dynamic processes of the underlying cyber-physical system. AoI thus serves as an appropriate metric that characterizes the trade-off between the congestion of the network and the timely generation of updates.

In this paper, we consider a scenario of spreading information in a wireless ad hoc network, where each node broadcasts its status updates from time to time and receives updates from others when not sending. Broadcast scheduling and power control have to be carefully designed to mitigate the interference caused by simultaneous transmissions over shared spectrum. The cross-layer design of scheduling and power allocation has been widely studied in terms of throughput maximization and found to be computationally complex [7]. In this paper, we consider online scheduling and power control jointly for AoI minimization based on the dynamics of AoI state and the SINR constraint which guarantees the quality of updates.

Wireless ad hoc networks have no static infrastructure or centralized coordination. The nature of peer-to-peer transmission and reception necessitates distributed protocols for broadcast scheduling and power control. Each individual node takes its action only based on the local observation of the overall system. We formulate a Markov game for this problem, where each agent aims to maximize its own utility. Due to the lack of an accurate system model at each node, we utilize a reinforcement learning framework to find the online policy. In [4], a reinforcement learning algorithm is used to learn unknown system parameters to minimize long-term average AoI. Here, we propose a multi-agent deep reinforcement learning algorithm. Deep reinforcement learning (DRL) is a powerful tool to design a dynamic control for the system with a large dimensional state space. It has become popular in solving challenging problems in game playing and robotics [8]. Recently, DRL has been highlighted by its applications in wireless networks. A growing number of papers have studied various wireless communication networks via DRL due to its capability of dealing with model-free and large dimensional systems. In [9], distributed dynamic spectrum access is studied via DRL and game theoretic analyses are given. Reference [10] studies transmit power control and interference mitigation in a large-scale network

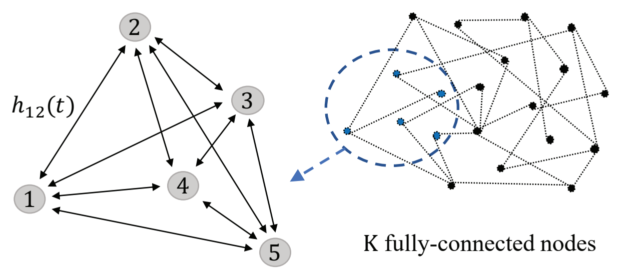

Fig. 1. System model.

via a multi-agent deep Q-learning algorithm, which provides near-optimal performance. In our proposed multi-agent DRL algorithm, recurrent neural network (RNN) is adopted to address the partially observable game. Simulation results verify its efficacy and remarkable performance improvement.

II. System Model

We investigate a wireless ad hoc network consisting of K fully-connected nodes as in Fig. 1. The system is time-slotted with slot length τ. Nodes, indexed by $k \in \mathcal{K} = \{1, 2, \ldots, K\}$, are synchronized over T slots. The nodes are equipped with half-duplex transceivers, i.e., each node is either transmitting or receiving in a slot. All share the same frequency band. Each node aims to communicate status information with all the others, i.e., it broadcasts its status updates to the other nodes and receives others' updates when not sending. Multiple nodes are allowed to transmit simultaneously. The receiving nodes decode the messages by successive interference cancellation. We assume all nodes can communicate with each other via a single hop, i.e., there are $K(K-1)/2$ bidirectional links.

We assume block fading channels. In slot t, $t = 1, 2, \ldots, T$, the channel gain between node k and j, $k \neq j$, is

$$h_{kj}(t) = \alpha_{kj}\beta_{kj}|\phi_{kj}(t)|^2, \tag{1}$$

where α_{kj} and β_{kj} denote the path loss and log-normal shadowing that are assumed to be constant over the session. $\phi_{kj}(t)$ is circularly symmetric complex Gaussian with zero mean and unit variance $\mathcal{CN}(0,1)$, i.e., we have Rayleigh fading. Similar to [10]–[12], we adopt Jakes model to characterize the temporal correlation between two consecutive slots,

$$\phi_{kj}(t) = \rho\phi_{kj}(t-1) + \sqrt{1-\rho^2}n(t), \tag{2}$$

where ρ denotes the correlation and $n(t) \sim \mathcal{CN}(0,1)$ is the independent randomness. The correlation is determined by $\rho = J_0(2\pi f_d \tau)$ with Bessel function of the first kind of order 0, J_0, and Doppler frequency f_d [13]. Let $p_k(t)$ denote the transmission power of node k at slot t. The signal-to-interference-plus-noise ratio (SINR) received at node j is given by

$$\Gamma_{kj}(t) = \frac{h_{kj}(t)p_k(t)}{\sum_{i \prec k} h_{ij}(t)p_i(t) + \sigma^2}, \tag{3}$$

where $i \prec k$ signifies that the received signal power of node i is smaller than that of node k and σ^2 is the received noise power.

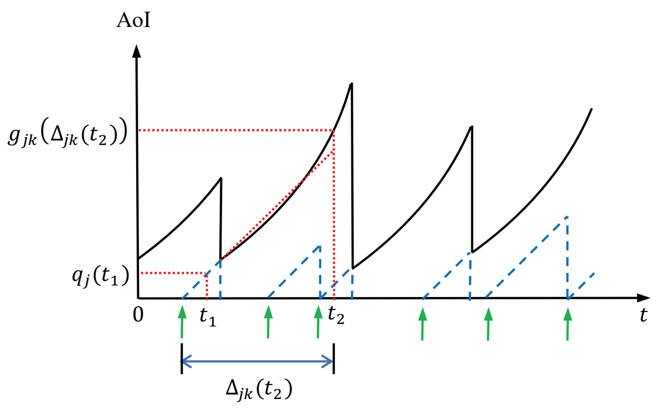

Fig. 2. AoI (infinitesimal τ). Solid black curves represent AoI of Rx at node k for updates from node j with function g_{jk}, blue dash lines represent AoI of Tx at node j, green arrows indicate generation instants of updates. The red dot lines illustrate the relationship of AoI of Tx and AoI of Rx. The second and fifth updates are overwritten by new updates before broadcasted.

At each node, update generation follows a stationary random process, whose distribution is generally unknown to the node. There is a buffer to store the generated data packet. The new packet overwrites the old one if it has not been broadcasted before the generation of the new one. Each transmission empties the buffer if no new packet is generated. Thus, only the newest data is transmitted, which is consistent with the requirement of information freshness in practical applications. In the following, we use "status update" and "data packet" interchangeably.

In most work up to date, a linear model of AoI is adopted, where AoI is defined as the time duration between the current time t and the generation instant of the most recent received packet by t. Specifically, the linear age of status updates from node j to node k is given by

$$\Delta_{jk}(t) \triangleq t - u_{jk}(t), \tag{4}$$

where $u_{jk}(t)$ is the generation time of the most recent received update by t. Here, we use the linear model to count the age of the packet in each node's buffer from a transmitter perspective, namely, AoI of Tx. We use $q_k(t)$ to denote the time elapsed since the generation of the stored packet at node k, where $q_k(t) = 0$ indicates there is no packet in the buffer. On the other hand, as a receiver (Rx), node k also records the age of the received updates from other $K-1$ nodes, namely, AoI of Rx, which is denoted by $A_{jk}(t)$ for updates from node j, $j \neq k$. We consider a general model for AoI at the Rx. Specifically, AoI of Rx for node k is given by $A_{jk}(t) \triangleq g_{jk}(\Delta_{jk}(t))$, where the function $g : [0, \infty) \mapsto [0, \infty)$ is assume to be measurable, non-negative, and non-decreasing [3]. For instance, g could be the power function $g(x) = x^c$, $c > 1$. This general age model is introduced to represent the dissatisfaction for data staleness and indicate the weights of status update for different links. We consider a general setup that the function g_{jk}, which is determined by the receiver k, is unknown at the associated transmitter j, but only $\Delta_{jk}(t)$ is recorded at j over time.

We assume that an update can be successfully delivered by the end of each slot if a received SINR constraint is satisfied [13]. As shown in Fig. 2, the AoI increases between successfully delivered updates and drops to the age of the update when an update is delivered. Specifically, with a successful update from transmitting node j, AoI of Rx at receiving node k decreases to $g_{jk}(q_j(t)+\tau)$ at the beginning of slot $t+1$, i.e., $\Delta_{jk}(t+1) = q_j(t)+\tau$. Otherwise, it increases to $g_{jk}(\Delta_{jk}(t+1))$. Thus, the AoIs of Tx and Rx for node k evolve as follows.

$$q_k(t+1) = \begin{cases} 0, & \text{if transmitting in slot } t \\ & \quad \text{or no packet arrives,} \\ q_k(t)+\tau, & \text{otherwise.} \end{cases} \tag{5}$$

$$A_{jk}(t+1) = \begin{cases} g_{jk}(q_j(t)+\tau), & \text{if } \Gamma_{jk}(t) \geq \Gamma_{\min}, \\ g_{jk}(\Delta_{jk}(t+1)), & \text{otherwise.} \end{cases} \tag{6}$$

After a slot, in which node k transmits an update, it receives a 1-bit feedback from each receiving node, indicating whether the packet is successfully delivered. By collecting feedback from all receiving nodes, the transmitting node can infer the transmitting/receiving status of all other nodes, which facilitates decisions in the slots onwards.

III. Problem Formulation

We aim to derive an online policy that dynamically schedules the broadcasting nodes and their transmission power to minimize the expected average AoI at each node. We consider a distributed setting without central coordination or system state information exchange to manage the scheduling. Each node can only partially observe its local state information. Thus, a partially observable Markov game [14], [15], is formulated. A Markov game for K agents is defined by a set of states $\mathcal{S}$ for all agents, a set of actions $\mathcal{P}_1, \dots, \mathcal{P}_K$ and a set of observations $\mathcal{O}_1, \dots, \mathcal{O}_K$ for each agent. Each agent chooses its action according to its policy $\pi_k : \mathcal{O}_k \mapsto \mathcal{P}_k$, which leads to a new state. The state transitition is according to the function $\mathcal{T} : \mathcal{S} \times \mathcal{P}_1 \times \cdots \times \mathcal{P}_K \mapsto \mathcal{S}$. After taking the action, agent k obtains a reward $r_k : \mathcal{S} \times \mathcal{P}_k \times \mathcal{P}_{-k} \mapsto \mathbb{R}$, which depends on other agents' actions $\mathcal{P}_{-k}$. Each agent receives a new observation associated with the new state: $o_k : \mathcal{S} \mapsto \mathcal{O}_k$, and attempts to maximize its expected return $R_k = \sum_{t=1}^{T} \gamma^{t-1} r_k(t)$ for time horizon T, where $\gamma \in [0,1)$ is a discount factor. Next, we define the components of the Markov game in our system.

Action: The action taken in slot t for node k is its transmission power $p_k(t) \in \mathcal{P}$, where $\mathcal{P}$ is a set of discrete power levels same for all nodes [11]:

$$\mathcal{P} \triangleq \Big\{0,\ P_{\min}\Big(\frac{P_{\max}}{P_{\min}}\Big)^{\frac{i}{|\mathcal{P}|-2}},\ i = 0, \dots, |\mathcal{P}|-2\Big\}, \tag{7}$$

where $P_{\min}$ and $P_{\max}$ are the minimum (non-zero) and maximum transmission power, respectively. In particular, $p_k(t) = 0$ indicates that the node is in receiving mode.

State: The system state, denoted by $s(t) \in \mathcal{S}$, consists of AoI of Tx, AoI of Rx, and channel gain for all nodes, i.e., $s(t) = (\mathbf{q}(t), \mathbf{A}(t), \mathbf{h}(t))$, where $\mathbf{q}(t)$ is a vector with entries $q_k(t)$, and $\mathbf{A}(t)$, $\mathbf{h}(t)$ are $K \times (K-1)$ matrices with entries $A_{kj}(t)$, $h_{kj}(t)$, $\forall j,k, j \neq k$, respectively.

Observations: Node k, $\forall k$, observes a local state of the system $o_k(t) \in \mathcal{O}_k$, which includes AoI of Tx $q_k(t)$, AoI of Rx associated with the updates from other nodes: $\mathbf{A}_{\_k}(t) \triangleq [A_{jk}(t)]_{j \neq k}$, linear age of packets from itself to other nodes: $\mathbf{\Delta}_{k\_}(t) \triangleq [\Delta_{kj}(t)]_{j \neq k}$, and the latest updated channel gain: $\tilde{\mathbf{h}}_k(t) = [\tilde{h}_{kj}(t)]_{j \neq k}$, obtained from the most recent transmission. Specifically, $\tilde{h}_{kj}(t) = h_{kj}(t_1)$, where t_1 is the most recent slot by t that an update is sent or received at node k or j. Node k can also infer from the 1-bit feedback signal the transmitting/receiving status of all nodes, denoted by $\mathbf{z}(t-1) = [z_k(t-1)]_{k \in \mathcal{K}}$, with $z_k(t-1) = 1$ indicating k transmits in slot $t-1$ and $z_k(t-1) = 0$ indicating k receives. Thus, the observation at time t is $o_k(t) = (q_k(t), \mathbf{A}_{\_k}(t), \mathbf{\Delta}_{k\_}(t), \tilde{\mathbf{h}}_k(t), \mathbf{z}(t-1))$.

Reward: The reward at each node is its age utility, which is defined as the negative sum of the AoI of Rx associated with its transmitted and received updates,

$$r_k(t) = -\sum_{j \neq k} \big(A_{kj}(t) + A_{jk}(t)\big). \tag{8}$$

The aim is to maximize the discounted expected return over an infinite horizon:

$$J_k = \mathbb{E}_{\pi_k}\Big[\sum_{t=1}^{\infty} \gamma^{t-1} r_k(t)\Big], \tag{9}$$

which can be used to approximated the following average return as $\gamma \to 1$ [16],

$$J_k = \limsup_{T \to \infty} \mathbb{E}_{\pi_k}\Big[\frac{1}{T}\sum_{t=1}^{T} r_k(t)\Big]. \tag{10}$$

The expectation is taken over policy π_k of node k. Without knowing the transition function of the system, each node learns its policy by deep reinforcement learning introduced next.

IV. Deep Reinforcement Learning Algorithm

In this section, we first give a brief background of Q-learning and deep Q-network (DQN) and then introduce the proposed multi-agent deep recurrent Q-learning algorithm.

A. Preliminaries

Q-learning is one of temporal-difference (TD) methods in reinforcement learning [17]. It is a model-free method that is able to estimate expected return by learning from the temporal difference of the value function without requiring prior knowledge of the system model. It has been widely applied in various decision-making problems. Q-learning approximates value function of optimal policy by the update

$$Q(s,a) \leftarrow (1-\eta)Q(s,a) + \eta\Big(r + \gamma \max_{a'} Q(s', a')\Big), \tag{11}$$

where s and a denote the state and action, respectively, $0 < \eta < 1$ is the learning rate. The policy is chosen as the one that maximizes the Q-value at each step.

Q-learning uses the table memory to save the Q-value of each state-action pair; this prevents the method from addressing high-dimensional problems. DRL, in general, is a combination of deep learning and reinforcement learning to deal with this curse of dimensionality. DQN proposed in [8] remarks the first success of such a combination. DQN evaluates the Q-value of all possible actions by taking advantage of function approximation of neural networks. An approximate value function $Q(s,a|\theta)$ is parameterized by θ, the weights of the neural network. To stabilize the learning process, a target network $\bar{Q}$ parameterized by $\bar{\theta}$ is employed and updated periodically from the estimation network Q. The weights of Q are updated by minimizing the following loss function in each step:

$$\mathcal{L}(\theta) = \mathbb{E}\Big[\Big(r + \gamma \max_{a'} \bar{Q}(s', a'|\bar{\theta}) - Q(s,a|\theta)\Big)^2\Big]. \tag{12}$$

Moreover, the generated samples are stored in a memory buffer and randomly sampled to feed the training process in each step, namely, experience replay.

B. Multi-Agent Deep Recurrent Q-Learning

In many real-world applications, the learning agents only partially observe the environment without complete state information but need to make the best decision at each step based on the partial observation. Multiple agents further complicate learning since interactions among the agents reshape the environment and each agent observes the outcomes not only led by its own action but also depending on the behavior of others [18]. To facilitate the agent to learn the true state of the system, we incorporate a recurrent neural network (RNN) structure, i.e., a long-short term memory (LSTM) layer into the network [19], [20]. An LSTM unit consists of a cell, an input gate, an output gate, and a forget gate. The cell tracks the dependencies of the elements in the input sequence, and the three gates control to which extent a value is fed into the cell, used to calculate the output, and to be retained in the cell, respectively. The LSTM structure helps the agents maintain an internal state and aggregate the history of observations, from which the agents can infer the system state. At each agent, we train a deep recurrent Q-network (DRQN) in parallel. Specifically, the input of the network at node k is a sliding window history including the actions and observations of the past M slots,

$$\mathcal{H}_k(t) \triangleq \Big([p_k(i)]_{i=t-M+1}^{t}, [o_k(i)]_{i=t-M+1}^{t}\Big). \tag{13}$$

The input is directly fed into a 2-layer LSTM of sequence length M, whose output goes into a hidden linear layer. Finally, a linear layer outputs the Q-values for $|\mathcal{P}|$ actions. A ε-greedy exploration policy is adopted to control the trade-off of exploration and exploitation. The action is selected by

$$p_k(t) = \begin{cases} \arg\max\limits_{p\in\mathcal{P}} Q_k\big(\mathcal{H}_k(t), p|\theta_k\big), & \text{w.p. } 1-\varepsilon, \\ \text{randomly select from } \mathcal{P}, & \text{w.p. } \varepsilon, \end{cases} \tag{14}$$

Algorithm 1 Multi-Agent Deep Recurrent Q-Learning

```
1:  Input episode number $N$, step number $T$, learning rate $\eta$,
    hard update rate $\mu$, exploration probability $\varepsilon$
2:  Initialize the estimation network $Q_k$ with random weights
    $\theta_k$, and initialize the target network $\bar{Q}_k$ by $\bar{\theta}_k \leftarrow \theta_k, \forall k$
3:  for episode $n = 1, \dots, N$ do
4:    Reset the simulation environment and receive initial
      state $s_k(1)$
5:    for step $t = 1, \dots, T$ do
6:      for node $k = 1, \dots, K$ do
7:        Select action $p_k(t)$ according to (14);
8:        Execute action $p_k(t)$, receive reward $r_k(t)$, and
          obtain the next state $\mathcal{H}_k(t+1)$;
9:      end for
10:     Store transition $\{\mathcal{H}_k(t), p_k(t), r_k(t), \mathcal{H}_k(t+1)\}_{k\in\mathcal{K}}$
        in the memory buffer;
11:     Randomly sample a mini-batch of $B$ transitions;
12:     Update the estimation network $Q_k$ by minimizing the
        loss in (15) with the mini-batch, $\forall k$;
13:     Update target network for every $\mu$ episodes.
14:   end for
15: end for
```

where ε is the exploration probability. At each step, a new transition $(\mathcal{H}_k(t), p_k(t), r_k(t), \mathcal{H}_k(t+1))$ is stored in the memory buffer and a mini-batch of B transitions $\{(\mathcal{H}_i, p_i, r_i, \mathcal{H}'_i)\}_{i=1}^{B}$ is randomly sampled from the buffer. By minimizing the following loss function, the weights of the estimation network Q are updated.

$$\mathcal{L}(\theta_k) = \frac{1}{B}\sum_{i=1}^{B}\Big(r_i + \max_{p\in\mathcal{P}} \bar{Q}_k\big(\mathcal{H}'_i, p|\bar{\theta}_k\big) - Q_k\big(\mathcal{H}_i, p_i|\theta_k\big)\Big)^2 \tag{15}$$

The target network $\bar{Q}$ is updated for every μ episodes. The algorithm is summarized in Algorithm 1.

V. Simulation

In the simulation, we consider networks for $K = 5$, 10, 20. The nodes locate uniformly and distributedly in a square. The parameters of the system setup are listed in Table. I. The path loss model is chosen according to the LTE standard for picocells [21]. For the structure of the Q-network, we have a two-layer LSTM with output size 256. The size of the hidden linear layer is 256×128. ReLU activation function and optimizer Adam are used [22]. The hyper-parameters for training are $N = 2000$ episodes, $T = 100$ steps (slots), learning rate $\eta =$ 1e-4, hard update rate $\mu = 10$, exploration probability $\varepsilon = 0.1$, batch size $B = 32$, discount factor $\gamma = 0.95$. The testing result is averaged over 100 runs with $T = 10000$ for each.

The proposed DRQN algorithm is compared with the round robin (RR) scheduler combined with two different power allocation methods, namely, RR-MaxPower, and RR-PowerControl. In round robin scheduling, nodes are scheduled to transmit in a fixed order. When it is the node's

TABLE I
SIMULATION PARAMETERS.

Carrier frequency	2 GHz
Bandwidth	1 MHz
Noise power	-174 dBm/Hz
Area size	100 m $\times$ 100 m
Path loss model	38 + 30 log $10(d)$, d in km
Log-normal shadowing	6 dB standard deviation
Doppler frequency	10 Hz
Slot length	10 ms
Maximum power	200 mW
Minimum power	1 mW
Power level	10

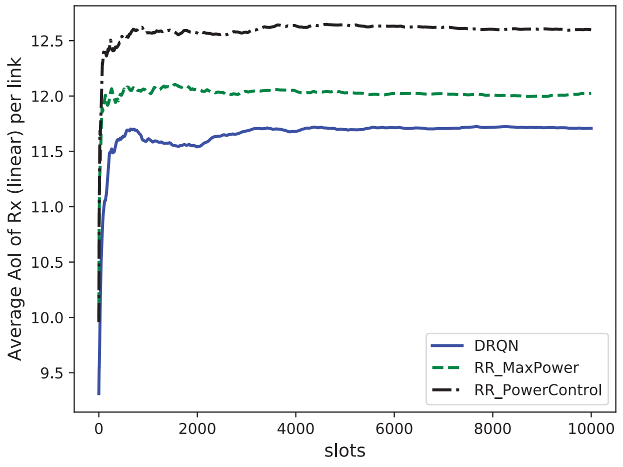

Fig. 3. Moving average linear AoI of Rx per link for $K = 5$ heterogeneous nodes.

turn, an update is sent if the node has a packet, otherwise, no one transmits. RR-MaxPower refers to when maximum power $P_{\max}$ is used in transmission. With RR-PowerControl, if a node is scheduled to transmit, it calculates its transmission power by minimizing the power consumption subject to the SINR constraint based on the local channel state information, and the closest larger power level is chosen. For the proposed DRQN, the node takes an action if it has a packet to transmit; if not, it can still receive. In all algorithms, each node only has the local information of channel gain that is updated since the last transmission or reception as aforementioned in Sec. III.

TABLE II
SETUP OF HETEROGENEOUS NODES

	$g_{jk}(\cdot)$	$\bar{d}$ (m)	κ	$\Gamma_{\min}$ (dB)	Return
Node 1	1 1 1 1	59.7	0.1	-3	-387.7
Node 2	1 0 0 0	35.7	0.05	0	-284.7
Node 3	1 1 0 0	34.5	0.2	0	-204.6
Node 4	0 0 1 0	54.9	0.1	3	-175.5
Node 5	1 0 1 0	36.4	0.2	-3	-148.4

In Fig. 3, we show the moving average linear AoI of Rx per link, i.e., $\frac{1}{tK(K-1)}\sum_{i=1}^{t}\sum_{k}\sum_{j\neq k}\Delta_{jk}(i)$, for a network with $K = 5$ heterogeneous nodes. The nodes have different functions g_{jk} for AoI of Rx, average distance to others $\bar{d}$, update generation probabilities κ, and SINR requirements $\Gamma_{\min}$. A detailed setup of the nodes is given in Table. II.

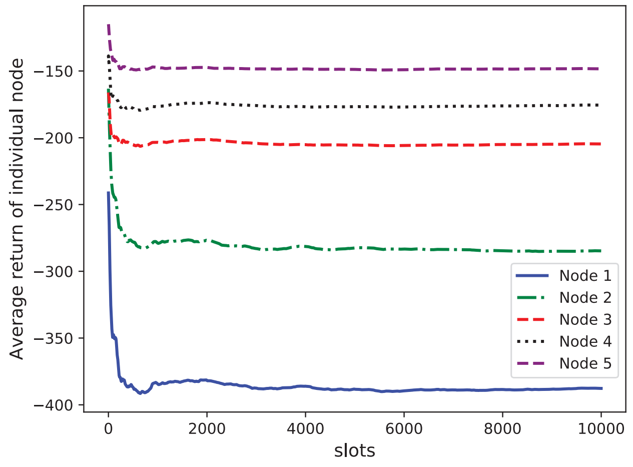

Fig. 4. Moving average reward of individual node for $K = 5$.

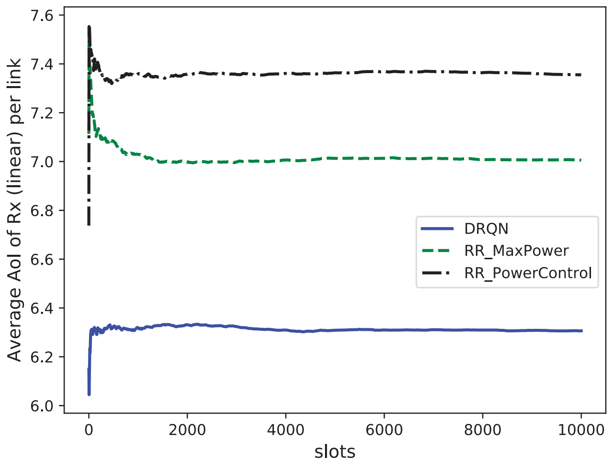

Fig. 5. Moving average linear AoI of Rx per link for $K = 5$ homogeneous nodes.

For AoI of Rx in each link, we either use identity function $g_{jk}(x) = x$ or power function $g_{jk}(x) = x^{1.5}$, denoted by 0 and 1 in Table. II, respectively. The update generation probability is i.i.d. over slots. As shown in Fig. 3, the proposed DRQN achieves a significantly smaller average AoI compared to the baselines. This verifies that the nodes learn from the local observations to take actions. The average return of individual node obtained in DRQN is plotted in Fig. 4. By the reward defined in (8), we observe that the average return of each node correctly reflects its circumstance of sending and receiving status updates.

Fig. 5, 6 and 7 illustrate the moving average linear AoI of Rx per link for different numbers of homogeneous nodes, $K = 5$, 10, 20, respectively. The update generation probability for each node is $\kappa = 0.2$ and the SINR requirement is $\Gamma_{\min} = 0$ dB. It can be observed that the proposed DRQN algorithm outperforms all baselines and shows significant superiority when the number of nodes is large so that more nodes are allowed to transmit simultaneously. In Table III, we list the average power consumption per transmission.

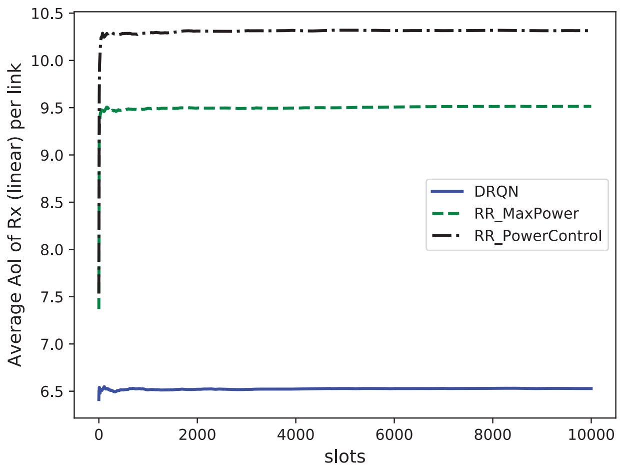

Fig. 6. Moving average linear AoI of Rx per link for $K = 10$ homogeneous nodes.

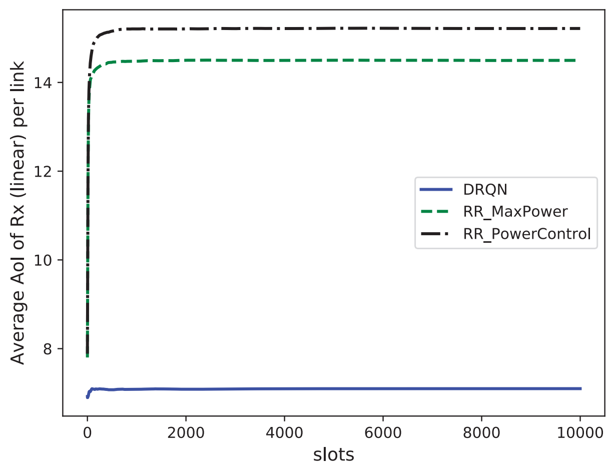

Fig. 7. Moving average linear AoI of Rx per link for $K = 20$ homogeneous nodes.

The proposed algorithm shows a remarkable energy-efficient advantage compared to RR-MaxPower and even a lower consumption than RR-PowerControl for $N = 20$.

TABLE III
AVERAGE TRANSMISSION POWER

	DRQN	RR-MaxPower	RR-PowerControl
$N = 5$	18.8 mW	134.4 mW	1.8 mW
$N = 10$	13.3 mW	178.4 mW	12.7 mW
$N = 20$	11.7 mW	197.6 mW	14.0 mW

VI. CONCLUSION

In this paper, we have studied AoI minimization in a wireless ad hoc network via DRL. With the objective of minimizing the expected average AoI of each node, we have formulated a Markov game. The framework of DRL has been used to derive an online policy for broadcasting scheduling and power control without knowing the system transition function. A multi-agent deep recurrent Q-learning algorithm has been proposed to address the partial observability. Simulation results have verified the proposed algorithm can be implemented distributedly with significant performance improvement over baselines. This study thus indicates that deep reinforcement learning is helpful in providing timely updates in an ad-hoc network in a distributed fashion. Future work may include extensions to large scale networks, and reward structures tailored to specific applications.

REFERENCES

[1] S. Kaul, R. Yates, and M. Gruteser, "Real-time status: How often should one update?" in *Proc. IEEE INFOCOM*, 2012, pp. 2731–2735.

[2] R. D. Yates and S. K. Kaul, "The age of information: Real-time status updating by multiple sources," *IEEE Trans. Inf. Theory*, vol. 65, no. 3, pp. 1807–1827, 2019.

[3] Y. Sun, E. Uysal-Biyikoglu, R. D. Yates, C. E. Koksal, and N. B. Shroff, "Update or wait: How to keep your data fresh," *IEEE Trans. Inf. Theory*, vol. 63, no. 11, pp. 7492–7508, 2017.

[4] E. T. Ceran, D. Gündüz, and A. György, "Average age of information with hybrid ARQ under a resource constraint," *IEEE Trans. Wireless Commun.*, vol. 18, no. 3, pp. 1900–1913, 2019.

[5] A. Arafa, J. Yang, S. Ulukus, and H. V. Poor, "Age-minimal transmission for energy harvesting sensors with finite batteries: Online policies," *arXiv:1806.07271*, 2018.

[6] S. Leng and A. Yener, "Age of information minimization for an energy harvesting cognitive radio," *IEEE Trans. Cognitive Commun. and Networking*, vol. 5, no. 2, pp. 427–439, 2019.

[7] A. Ephremides and T. V. Truong, "Scheduling broadcasts in multihop radio networks," *IEEE Trans. Commun.*, vol. 38, no. 4, pp. 456–460, 1990.

[8] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[9] O. Naparstek and K. Cohen, "Deep multi-user reinforcement learning for distributed dynamic spectrum access," *IEEE Trans. Wireless Commun.*, vol. 18, no. 1, pp. 310–323, 2019.

[10] Y. S. Nasir and D. Guo, "Deep reinforcement learning for distributed dynamic power allocation in wireless networks," *arXiv preprint arXiv:1808.00490*, 2018.

[11] F. Meng, P. Chen, L. Wu, and J. Cheng, "Power allocation in multi-user cellular networks: Deep reinforcement learning approaches," *arXiv:1901.07159*, 2019.

[12] Z. Chen and X. Wang, "Decentralized computation offloading for multi-user mobile edge computing: A deep reinforcement learning approach," *arXiv:1812.07394*, 2018.

[13] A. Goldsmith, *Wireless communications*. Cambridge university press, 2005.

[14] M. L. Littman, "Markov games as a framework for multi-agent reinforcement learning," in *Machine learning proceedings*. Elsevier, 1994, pp. 157–163.

[15] R. Lowe, Y. Wu, A. Tamar, J. Harb, O. P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative competitive environments," in *Advances in Neural Information Processing Systems*, 2017, pp. 6379–6390.

[16] E. Fernández-Gaucherand, A. Arapostathis, and S. I. Marcus, "On the average cost optimality equation and the structure of optimal policies for partially observable markov decision processes," *Annals of Operations Research*, vol. 29, no. 1, pp. 439–469, 1991.

[17] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[18] T. T. Nguyen, N. D. Nguyen, and S. Nahavandi, "Deep reinforcement learning for multi-agent systems: A review of challenges, solutions and applications," *arXiv:1812.11794*, 2018.

[19] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[20] M. Hausknecht and P. Stone, "Deep recurrent Q-learning for partially observable MDPs," in *AAAI Fall Symposium Series*, 2015.

[21] Evolved Universal Terrestrial Radio Access, "Radio frequency (RF) requirements for LTE pico node B (release 15)," 2018.

[22] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv:1412.6980*, 2014.