Application-Oriented Scheduling for Optimizing the Age of Correlated Information: A Deep-Reinforcement-Learning-Based Approach

Bo Yin, *Student Member, IEEE*, Shuai Zhang, *Student Member, IEEE*, and Yu Cheng, *Senior Member, IEEE*

***Abstract*—Recent advances in communications technologies and the proliferation of connected devices have spawned a variety of information-centric Internet-of-Things (IoT) systems, where timely information updating is normally required. Age of Information (AoI) has recently been introduced to quantify the freshness of the knowledge the controller has about the remote information sources. With the development of IoT applications, it is becoming increasingly common that the application-level services, e.g., status updates, rely on the timely delivery of fresh information from a number of information sources. In this article, we study the application-oriented scheduling for optimizing information freshness in the presence of correlated information sources. To this end, we adopt the concept of Age of Correlated Information (AoCI) and formulate the scheduling problem as an episodic Markov decision process (MDP) problem with an application-oriented policy. Given the complexity of the above problem, we develop a learning-based approach that not only leverages the emerging deep reinforcement learning (DRL) techniques but also exploits diverse-domain knowledge. The numerical results show that the proposed approach achieves better performance in terms of AoCI, compared to some typical baseline methods.**

***Index Terms*—Age of Information (AoI), deep reinforcement learning (DRL), information correlations, scheduling.**

I. Introduction

THE Internet of Things (IoT) is regarded as one of the biggest paradigm shifts that have the potential to change all aspects of our lives drastically. Due to the convergence of pervasive connectivity and ubiquitous computing, IoT systems are becoming increasingly information centric, evolving from end-to-end data communication networks to integral intelligent systems that involve sensing, data analytics, information extraction, and decision making. In these systems, there is a commonly centralized controller that collects the raw data from diverse devices with different sizes and shapes and supports a variety of IoT applications, such as home security applications and home appliances management applications in a smart home ecosystem. The Quality of Service (QoS) of most of IoT applications, particularly those real-time ones, depends heavily on the timely delivery of fresh information. For example, in an industrial automation application, the control center needs to collect time-sensitive measurements from different machines for maintaining situational awareness and performing real-time control.

Given the importance of optimizing the timeliness of information update in real-time applications, a performance metric, called Age of Information (AoI), has recently been introduced to quantify the freshness of the knowledge the controller has about the remote information sources [1]. Specifically, the AoI is defined as the time elapsed since the generation of the latest updated information. Due to the capability of capturing information freshness in various applications, AoI has sparked tremendous interests and been studied in many networking systems [2], ranging from wireless sensor networks, cellular networks to vehicular networks. The optimization of AoI requires careful considerations over the whole lifetime of the information, i.e., from its creation to the delivery, which yields a series of novel system designs in the areas of sampling strategy [3]–[5], queuing discipline [6]–[8], and link scheduling policy [9]–[18].

The fundamental difference between AoI and other QoS metrics, e.g., latency and throughput, is that AoI is information centric rather than transmission centric. However, in the present state of the art, the evaluation of information freshness is still constrained at the packet level or message level, that is, the controller's knowledge gets refreshed upon the delivery of a single packet. In this article, we argue that the received information is effective only when it helps the applications or users to make certain meaningful decisions or facilitate specific application-level actions. From this angle, the existing packet-level AoI modeling cannot well meet the application-level service requirements in many practical scenarios, where a user or an application has to aggregate the information from multiple sources for making a sensible decision. For example, an indoor routing algorithm may rely on the real-time position information of the object, which is obtained by aggregating the range measurements from multiple anchor nodes. Generally, we say two information sources are correlated if information from them contributes to the same decision-making process. Given the pervasiveness of the information correlation, it is of importance to understand its influences on the development of freshness optimization approach. Nevertheless, few of the existing literature in this area seriously considered the correlations across different information sources.

Manuscript received January 28, 2020; revised April 14, 2020; accepted May 15, 2020. Date of publication May 21, 2020; date of current version September 15, 2020. This work was supported in part by the U.S. National Science Foundation, under Grant CNS-1816908 and Grant ECCS-1610874. *(Corresponding author: Yu Cheng.)*

The authors are with the Department of Electrical and Computer Engineering, Illinois Institute of Technology, Chicago, IL 60616 USA (e-mail: byin@hawk.iit.edu; szhang104@hawk.iit.edu; cheng@iit.edu).

Digital Object Identifier 10.1109/JIOT.2020.2996562

In this article, we target a study of application-oriented scheduling for optimizing information freshness in the presence of correlated information sources. Specifically, we consider a general IoT system, where various time-sensitive information monitored by different IoT devices is collected regularly to support multiple applications and study the transmission scheduling problem for this system. We adopt the concept of the Age of Correlated Information (AoCI) introduced in [19] to characterize the application-level freshness of information. Motivated by the recent successful applications of the deep reinforcement learning (DRL) framework in network optimization, we seek a DRL-based approach to deal with the correlations that arise in the scheduling problem. The main contributions of this article can be summarized as follows.

1) By leveraging the dynamics of the AoCI, we formulate the scheduling problem as an episodic Markov decision process (MDP) problem and prove its NP-hardness. Furthermore, we employ the options framework to model the scheduling policy, which incorporates the all-or-nothing characteristic of the AoCI-based status update.
2) We propose a DRL-based approach that can exploit a variety of domain knowledge and yield a reasonable scheduling policy. Specifically, we utilize the reward decomposition technique to exploit the special structure of the reward signal to reduce the complexities of the target approximations. Besides, an attention-integrated relevance network (ARN) architecture is developed for capturing the correlations among the applications.
3) We conduct extensive simulations to demonstrate the effectiveness of the proposed approach. The results show that a stable and reasonable approximation to the option-value function can be derived by the learning agent, which generates a scheduling policy that outperforms conventional algorithms.

The remainder of this article is organized as follows. Section II describes the system model as well as the freshness optimization problem in which the correlations across different information are considered. Section III introduces some basics of the RL framework and the emerging DRL techniques. The characterizations of the application-oriented scheduling policy are discussed in Section IV while Section V presents the proposed DRL-based approach. The performance evaluation of the proposed approach is provided in Section VI. Section VII overviews the related work. Finally, Section VIII concludes this article.

II. System Model and Problem Statement

A. System Model

We consider an IoT uplink system that is served by a local area network (LAN), e.g., the smart homes or Industrial IoT (IIoT) systems. An IoT hub, acting as the central controller, collects time-sensitive information from a set $\mathcal{M}$ of M IoT devices to support N IoT applications. The system works in a frame-based manner [13], [17], where each frame consists of T transmission slots and $T < M$. At the beginning of

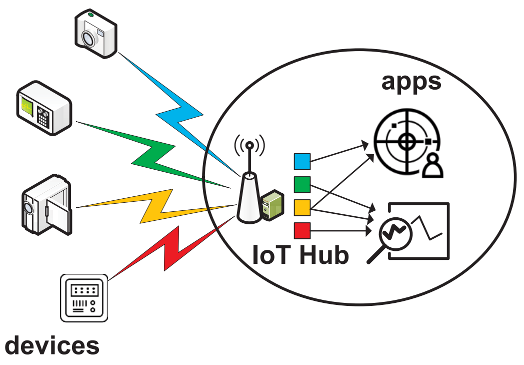


Fig. 1. IoT system with correlated information sources (the square with a certain color indicates a collected updating message from the corresponding source while the arrow means that the message is consumed by an application).

every frame, each device generates a packet[1] that contains the information of its current status. This synchronized update model occurs in many monitoring and control applications, where packets are generated periodically. The hub pulls the information from some devices during a transmission frame. At the end of that frame, the hub will update the statuses of the applications according to the information it collects. As illustrated in Fig. 1, we consider a generic scenario in which updating the status of an application requires to integrate the information gathered from multiple devices. Formally, let $\mathcal{F}_n \subseteq \mathcal{M}$ denote the set of devices whose information is necessary to update the status of application n. Moreover, the sets of devices that are associated with different applications can overlap. In other words, the packet from one device may be useful in updating multiple applications.

1) Transmission Model: The IoT hub is in charge of the packet transmission within each frame such that at most one device can receive the transmission grant in one slot. The channels between the devices and the hub are assumed to be unreliable. Specifically, transmission failures are modeled by independent and identically distributed Bernoulli processes, e.g., packet erasure channels in wireless networks. That is, if device m is scheduled in a slot, the probability for successful packet reception and decoding is p_m. In addition, the channels are assumed to be quasistatic. In this article, the channel characteristics are described for definiteness and we consider that the statistics of transmission failure are unknown. Packets that remain in the buffer, either undelivered or from unselected devices, will be dropped at the end of each frame.

2) Age of Correlated Information Model: We assume that the IoT hub can only utilize the messages which at generated at the same time to update the status of an application. In other words, the status of application n will not be updated until the latest packets from all devices in $\mathcal{F}_n$ are received by the hub. As considered in [19], we employ a coflow-like abstraction [20] to model the correlations across the information from multiple devices. Let $c_n(k, t)$ denote the AoCI with respect to application n at the beginning of the tth slot of frame k.

[1]The proposed approach can be extended to handle more general case, where one or multiple packets are needed to convey the information of a device, by encoding the transmission progress of each device with the unary coding scheme.

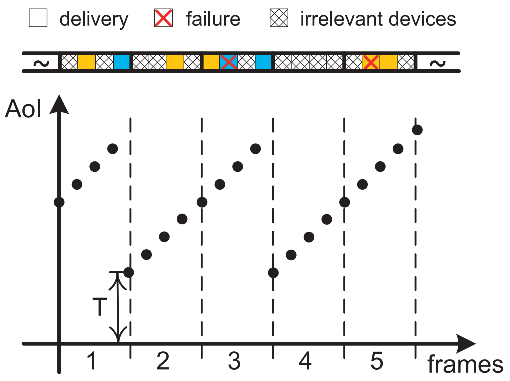


Fig. 2. AoCI dynamic of an application over five consecutive frames (the update for this application requires the packets from two devices, illustrated as blue and yellow squares).

Before update, $c_n(k, t)$ grows linearly in time. If all the packets required for updating the status of application n are delivered over the duration of a frame, say frame k, $c_n(k+1, 1)$ will drop to T since all these packets are generated at the beginning of frame k. The typical AoCI dynamic of an application over five consecutive frames is illustrated in Fig. 2. The status of this application is updated successfully at the end of the first and third frames.

B. Problem Statement

We study the scheduling policy for the aforementioned model. Let $a(k, t) \in \{0, 1, \ldots, M\}$ denote the scheduling decision at the tth slot of frame k, i.e., $a(k, t) = i, i > 0$ if device i is scheduled at that slot while $a(k, t) = 0$ indicates an idle slot. The quality of a scheduling policy π is characterized by the expected long-term average AoCI of all the applications in the IoT system, denoted by $J(\pi)$

$$J(\pi) = \limsup_{K\to\infty} \frac{1}{KNT}\mathbb{E}_\pi\left[\sum_{k=1}^{K}\sum_{n=1}^{N}\sum_{t=1}^{T} c_n(k, t)\right]. \quad (1)$$

During frame k, let $h_{n,k}$ denote the number of frames elapsed since the generation of the latest received packet of application n. The evolution of $h_{n,k}$ can be expressed as

$$h_{n,k+1} = \begin{cases} 1, & \text{if } \mathcal{F}_n \subseteq \mathcal{D}_k \\ h_{n,k} + 1, & \text{otherwise} \end{cases} \quad (2)$$

where $\mathcal{D}_k$ represents the set of devices whose packet gets delivered during frame k. Consider that the updates can only occur at the end of a frame, we have

$$c_n(k, t) = h_{n,k}T + t - 1. \quad (3)$$

With proper manipulations, $J(\pi)$ can be expressed as

$$J(\pi) = \limsup_{K\to\infty} \frac{T}{KN}\mathbb{E}_\pi\left[\sum_{k=1}^{K}\sum_{n=1}^{N} h_{n,k}\right] + \frac{T-1}{2}. \quad (4)$$

For a given system setting, i.e., N and T are fixed, the optimal policy that minimizes $J(\pi)$ is also the policy that achieves the lowest $\tilde{J}(\pi)$

$$\tilde{J}(\pi) = \limsup_{K\to\infty} \frac{1}{K}\sum_{k=1}^{K}\mathbb{E}_\pi\left[\sum_{n=1}^{N} h_{n,k}\right]. \quad (5)$$

In this article, we aim to leverage the RL framework to propose a scheduling policy that results in low $\tilde{J}(\pi)$. Particularly, we focus on the model-free approach, where the policy is derived through the interactions with the environment without estimating the dynamics of the environment, i.e., the probabilities p_m. The model-free approach can be easily adapted for the systems in which the environment is nonstationary, e.g., p_m changes slowly. Inspired by the recent advances in DRL techniques, we explore an experience-driven approach in which artificial neural networks (ANNs) are utilized as function approximators to extract valuable knowledge that is valuable for AoCI-based scheduling. In order to exploit both the spatial and temporal structures of the scheduling problem, we will develop a multiscale hierarchical framework. More precisely, the learning agent makes the slot-level scheduling decisions under the guidance of a frame-level policy while the per-slot decision is abstracted as the outcome of a two-tier decision-making process.

III. Preliminaries

In this section, we introduce some basics of the RL framework as well as the emerging DRL techniques. Generally, the RL methods are applied to solve an MDP in which an agent interacts with an environment in a discrete decision slot. Let $\mathcal{S}$ and $\mathcal{A}$, respectively, denote the state space of the environment and the action space of the agent. At slot τ, the agent observes the current state of the environment, say $S(\tau)$, and takes an action $A(\tau)$ according to a policy π, which is a mapping from $\mathcal{S}$ to a probability distribution over $\mathcal{A}$. Afterward, the state of the environment becomes $S(\tau+1)$ and the agent receives a scalar reward $R(\tau)$, which is determined by the reward function $r(S(\tau), A(\tau), S(\tau+1))$. Let $G(\tau)$ denote the discounted cumulative reward, i.e., $G(\tau) = \sum_{t=\tau}^{\infty}\gamma^{t-\tau}R(t)$, where $\gamma \in [0, 1]$ is the discount factor. The goal of the agent is to derive the optimal policy that maximizes $G(\tau)$ in the sense of expectation.

The action-value function for policy π, denoted by $Q^\pi(s, a)$, plays an important role in model-free RL methods. It quantifies the value of taking action a in state s under π, i.e.,

$$Q^\pi(s, a) = \mathbb{E}_\pi[G(\tau)|S(\tau) = s, A(\tau) = a]. \quad (6)$$

A fundamental property of the action-value function for any policy π is that it satisfies the following recursive property, also known as, the Bellman equation:

$$Q^\pi(s, a) = \mathbb{E}_{s'}\big[R(\tau) + \gamma\mathbb{E}_{a'\sim\pi(s')}\big[Q^\pi(s', a')\big]\big] \quad (7)$$

where s' is the next state and a' is the next action. This property is used throughout the RL methodology. Basically, if the action-value function for an optimal policy is known, say $Q^*(s, a)$, then the policy that acting greedily with respect to $Q^*(s, a)$, i.e., $A(\tau) = \text{argmax}_a Q^*(S(\tau), a)$, is also optimal. The famous Q-learning algorithm [21] leverages the Bellman optimality equation and improves the estimate of $Q^*(s, a)$ iteratively via sample-based update as follows:

$$Q(S(\tau), A(\tau)) := Q(S(\tau), A(\tau)) + \alpha\delta(\tau) \quad (8)$$

where α is the learning rate and $\delta(\tau)$ is called temporal-difference (TD) error and can be calculated as

$$\delta(\tau) = R(\tau) + \gamma \max_a Q(S(\tau+1), a) - Q(S(\tau), A(\tau)). \quad (9)$$

When the number of the state–action pair is enormous, ANNs are used by DRL methods for learning a low-dimensional representation of the estimate of $Q^*(s,a)$. In this way, the action-value function is expressed as a parameterized functional form with weight vector $\boldsymbol{\theta}$, denoted by $Q(s,a;\boldsymbol{\theta})$. Typically, the ANN of $Q(s,a;\boldsymbol{\theta})$, called deep Q-network (DQN), takes the state representation as its input and outputs a $|\mathcal{A}|$-dimensional vector, each component of which is the Q-value with respect to a specific action. The weights $\boldsymbol{\theta}$ are updated iteratively with the objective of minimizing a loss function $\mathcal{L}(\theta)$ via the gradient-based method $\boldsymbol{\theta} := \boldsymbol{\theta} - \alpha\Delta_{\boldsymbol{\theta}}\mathcal{L}(\theta)$, i.e., backpropagation algorithms. The main challenge of the DRL methods is that the ANNs diverge aggressively during the training process. The DQN approach in [22] introduces several novel techniques to improve stability. First, a separate target network, denoted by $\tilde{Q}(s,a;\boldsymbol{\theta}^-)$ is used to calculate the TD errors. The target network is a copy of the DQN except that its weights $\boldsymbol{\theta}^-$ are updated less frequently. More precisely, the target network duplicates the current weights of the DQN regularly whenever a certain number of updates of $\boldsymbol{\theta}$ have been performed. $\boldsymbol{\theta}^-$ are held fixed during this interval. Second, the agent stores its experience, say (s,a,r,s'), in a replay buffer. Instead of the latest sample in the relay buffer, a minibatch of samples that are randomly drawn from the replay buffer is used to update $\boldsymbol{\theta}$, which eliminates the correlation between successive updates and smooths out the training. In this way, the loss function that quantifies the expected TD errors is expressed as

$$\mathcal{L}(\boldsymbol{\theta}) = \mathbb{E}_{(s,a,r,s')}\Big[(y - Q(s,a;\boldsymbol{\theta}))^2\Big] \quad (10)$$

where $y = r + \gamma \max_{a'} \tilde{Q}(s',a';\boldsymbol{\theta}^-)$ is the training target. In addition, clipping techniques, e.g., reward clipping and error clipping, are used to further mitigate the divergence issue.

The works in [23]–[25] also shed light on the development of the DRL-based scheduling policy. Specifically, the work in [23] proposes the options framework to enable temporal abstraction and boost exploration in RL. As a generalization of primitive actions, an option includes temporally extended actions. Formally, an option o is denoted by a tuple $(\mathcal{I}_o, \phi_o, \beta_o)$ in which $\mathcal{I}_o$ is an initiation set, ϕ_o is called option's policy (intraoption policy), and β_o is a termination condition. Specifically, if the option o is initiated in state $s \in \mathcal{I}_o$, actions will be taken according to ϕ_o until the termination condition β_o is met. The selection of options is controlled by policy over options (interoption policy), denoted by μ. Commonly, the initiation set of option o is defined as $\{s : \beta_o(s) = 0\}$. Thinking of options as the subgoals of the system, an option's policy is designed to achieve that specific subgoal.

The method of hybrid reward architecture (HRA) proposed in [24] facilitates the DQN training with decomposed reward signals. The key insight of the HRA method is that rather than an accurate approximation, a sufficiently consistent approximation to the Q-value function is adequate for deriving a reasonable policy. A function $Q(s,a;\boldsymbol{\theta})$ is said to be consistent with $Q^*(s,a)$ if acting greedily with respect to it also results in an optimal policy. A straightforward example is a product of $Q^*(s,a)$ and a constant number, noting that the relative rank of the Q-values of different actions with respect to an arbitrary state does not change. In this sense, acting greedily with respect to a semiconsistent approximation results in a good policy. Furthermore, the results in [24] show that a semiconsistent approximation can be obtained by employing a uniformly random policy.

For handling the representations learning in text-based games, a deep reinforcement relevance network (DRRN) is proposed in [25] to capture the semantic and syntactic information in the state space and action space, which are both described by the text. Different from the ANN architecture in the regular DQN approach that directly outputs the Q-values with respect to particular actions, DRRN models the Q-values via an interaction function, which characterizes the relevance between the high-level embedding vectors of the specific state and action.

IV. Options-Based Modeling

A. MDP Formulation

With the objective (5), the scheduling problem can be naturally cast as an infinite horizon control problem. This innocent-looking formulation, nevertheless, could stifle the development of the scheduling policy. In the frame-based model, each device generates a new message periodically, which implies that the transmission states about the latest messages will be reset to a specific starting state at the beginning of every frame regardless of the scheduling policy. Such a recurring pattern makes it intractable to learn an ergodic policy, which is typically desirable for performing infinite horizon control. From a different perspective, the aforementioned unconditional transition can also be interpreted as the termination in an episodic task that interrupts the normal flow of state transitions. In this way, we resort to the episodic MDP model to develop the scheduling policy under which the learning agent will perform episodic control over the schedule of the transmissions. Specifically, the duty of the agent is to derive an intraframe control policy that could drive the system to the desirable states at the end of each frame. Exploiting this special structure of the scheduling problem reduces the complexity of learning a reasonable policy since the learning agent can restrict its attention to addressing the intricacies caused by the correlations between the information updates within a frame. To this end, a macropolicy is expected, which can make use of the frame-based dynamics of the system and guide the learning agent, e.g., informing the agent of what kind of states are favorable at the beginning of each frame.

Consider the similarity between the dynamics of $h_{n,k}$ and that of a network queue, we employ the Max-Weight policy [26], which is a widely used scheduling algorithm for networks of queues, as the macropolicy for the intraframe control. The Max-Weight policy can be derived via the Lyapunov optimization [27], [28]. Specifically, a quadratic Lyapunov function is used to quantify the desirability of the system states, i.e.,

$$L(\mathbf{h}_k) = \sum_{n=1}^{N} h_{n,k}^2 \tag{11}$$

where $\mathbf{h}_k = [h_{1,k}, h_{2,k}, \ldots, h_{N,k}]$. Intuitively, to achieve a low $\tilde{J}(\pi)$, the scheduling policy is expected to prevent $L(\mathbf{h}_{k+1})$ from becoming too large. Given $\mathbf{h}_k$, the Max-Weight policy achieves this by minimizing the Lyapunov drift, i.e.,

$$\Delta L(\mathbf{h}_k) = \mathbb{E}_\pi\big[L(\mathbf{h}_{k+1}) - L(\mathbf{h}_k)|\mathbf{h}_k\big]. \tag{12}$$

According to the results in [13], the Max-Weight policy in the frame-based setting is equivalent to the policy that maximizes the one-frame return, denoted by $G_k(\pi)$, at each frame

$$G_k(\pi) = \mathbb{E}_\pi\left[\sum_{n=1}^{N} \mathbb{1}_{\{\mathcal{F}_n \subseteq \mathcal{D}_k\}} h_{n,k}\big(h_{n,k}+2\big)\right] \tag{13}$$

where $\mathbb{1}_{\{e\}}$ is the indicator function which equals to 1 if the expression e is true.

Informally speaking, the Max-Weight policy breaks down the long-term goal (5) into frame-by-frame subgoals (13). Within a frame, it schedules the message transmissions with the objective of maximizing the sum of the "weights" of the applications whose status can be updated at the end of that frame. The "weight" of application n at frame k is $h_{n,k}(h_{n,k}+2)$. As shown in the following theorem, performing such episodic control in the presence of correlated information sources is generally a nontrivial task.

Theorem 1: The episodic control problem, i.e., maximizing $G_k(\pi)$ given $\mathbf{h}_k$, is NP-hard.

Proof: The NP-hardness of the episodic control problem is proven via reduction. Consider a special case of this problem where the channels are error free. Correspondingly, the episodic control problem becomes the problem of selecting at most T devices to transmit their messages such that $G_k(\pi)$ is maximized. Note that the relations between devices and applications can be characterized by a hypergraph, where each edge is denoted by an arbitrary set of nodes. Specifically, the nodes represent devices and the edges represent applications, each of which can be unequivocally identified by a set of devices, say $\mathcal{F}_n$. Besides, each edge is associated with a weight, say $h_{n,k}(h_{n,k}+2)$. In this way, the above problem is equivalent to finding the weighted densest subgraph with at most T nodes, which is the generalized version of the densest subgraph problem [29]. Due to the NP-hardness of the densest subgraph problem, the episodic control problem is NP-hard. ■

As discussed above, the episodic control framework enables the agent to explore the characterizations of the scheduling policy by considering only a single frame. Therefore, hereafter in this article, the frame index k is dropped for brevity. The discounted factor γ is also omitted since it equals to 1 in the episodic scenario. In order to apply the RL framework to solve the episodic control problem, we describe the state space, action space, and reward signal as follows.

1) *State Space:* A state captures the AoCI of each application and the transmission status with respect to the packet of each device. To ensure stationary state transition, the current slot index is also included. That is, $s \triangleq [h_1, h_2, \ldots, h_N; u_1, u_2, \ldots, u_M; t]$, where $u_m \in \{0, 1\}$ indicates whether the information from device i is delivered. At the beginning of a frame, $u_m = 0 \forall m$.
2) *Action Space:* The actions in the episodic control problem are the scheduling decisions of each slot. Therefore, $\mathcal{A} \triangleq \{0, 1, \ldots, M\}$.
3) *Reward Function:* Let d_n and d'_n be the indicator variables that, respectively, denote whether $\mathcal{F}_n \subseteq \mathcal{D}$ at the beginning and end of the current slot.[2] According to (13), the per-slot reward can be defined as $R = \sum_{n=1}^{N} r_n$, where

$$r_n = \begin{cases} d'_n h_n(h_n+2), & \text{if } d_n = 0 \\ 0, & \text{otherwise.} \end{cases} \tag{14}$$

B. Application-Oriented Scheduling

In general, an RL agent improves its control policy by exploring the action space stochastically and exploiting what it has learned from the reward signal. Due to the all-or-nothing nature of the status update of each application, the agent rarely receives positive rewards if the devices are scheduled in an arbitrary stochastic manner. In other words, straightforward exploration over the primitive action space will yield an extremely sparse reward signal. Under such a circumstance, it is very difficult for the RL agent to make progress since zero reward offers little knowledge about how to accumulate the rewards. In light of the hierarchical structure of the scheduling problem, we employ the options framework [23] to incorporate the temporally abstract knowledge and action, i.e., the status update of an application requires series of successful transmissions of messages from multiple devices. This allows the agent to efficiently explore the environment and learn a reasonable policy from the perspective of high-level abstract actions.

In the context of transmission scheduling, an option can be naturally associated with an application. Therefore, the option space is defined as $\mathcal{O} \triangleq \{0, 1, \ldots, N\}$, where $o = 0$ represents an idle slot. The scheduling policy π is the composition of the interoption policy and a collection of options' policies. For an option $o = n$, its termination condition is either all the packets application n needs are delivered or the remaining slots are not sufficient for transmitting the undelivered packets, i.e.,

$$\beta_n(s) = \mathbb{1}_{\left\{\left(\sum_{m\in\mathcal{F}_n}(1-u_m)\right)=0\right\}} \vee \mathbb{1}_{\left\{\left(\sum_{m\in\mathcal{F}_n} u_m\right)>T-t+1\right\}}. \tag{15}$$

In this article, we restrict our attention to scheduling policies in which the decision of idle slot can be made if and only if all the other options are terminated, i.e., $\beta_0(s) = 1 - \bigwedge_{n=1}^{N} \beta_n(s)$. It is not difficult to see that an arbitrary policy is dominated by at least one scheduling policy of this type. Therefore, the learning agent only considers the options $o \in \{1, \ldots, N\}$ in the development of the policies μ and $\{\phi_n\}$. In what follows,

[2]The values of d_n and d'_n can be derived from the current state s and the next state s', respectively.

indexes o and n are used interchangeably. Consider that option n implies the subgoal as updating the status of application n, it is not difficult to see that the corresponding option's policy is to sequentially schedule the devices in $\mathcal{F}_n$ whose information is not delivered yet.

Analogous to the action-value function, the option-value function for a policy μ quantifies the expected cumulative rewards the agent can obtain by initiating an option at a specific state, i.e.,

$$Q^{\mu}(s_t, o) = \mathbb{E}\left[\sum_{\tau=t}^{T} R(\tau) \middle| \mathcal{E}(o \circ \mu, s_t, t)\right] \tag{16}$$

where $\mathcal{E}(o \circ \mu, s_t, t)$ represents the events of following the composition of the intraoption policy of option o and the interoption policy. That is, keep executing the option o until its termination and then select another option according to the policy μ. With deterministic options' policies, Q-learning algorithm can be extended to the options-based formulation by leveraging the Bellman-like optimality equations [23]

$$Q^*(s, o) = \mathbb{E}\big[R + U^*\big(s', o\big)\big| s, \phi_o(s)\big] \tag{17}$$

where $U^*(s, o)$ is called the option-value function *upon arrival*, i.e.,

$$U^*(s, o) = (1 - \beta_o(s))Q^*(s, o) + \beta_o(s) \max_{o' \in \mathcal{O}} Q^*(s, o'). \tag{18}$$

The one-step TD update of the option-value function is manifested as

$$Q(s_t, o) := Q(s_t, o) + \alpha\big[R_{t+1} + U(s_{t+1}, o) - Q(s_t, o)\big] \tag{19}$$

where $U(s, o)$ represents the current estimate of $U^*(s, o)$.

The above algorithm can efficiently utilize the experience sample by updating the Q-values with respect to multiple options simultaneously. That is, the update rule (19) is applied to all those options that will take the same action a_t as that in the sample. With this in mind, an option's policy, say ϕ_n, is designed to schedule the devices $\{m : m \in \mathcal{F}_n\}$ in descending order of l_m, where l_m denotes the number of applications that require the information of device m. The establishment of such policies follows the intuition that scheduling devices with higher l_m earlier can in some sense increase the flexibility of intraoption learning. In the next section, we develop a DQN-based approach to learn an approximation to the optimal option-value function $Q^*(s, o)$ under the aforementioned options' policies and then derive the application-oriented scheduling policy.

V. DRL-Based Approach

A. Reward Decomposition

The options-based modeling captures the hierarchical structure of the scheduling policy such that the learning agent can explore the action space effectively. Unless extra care is taken, however, the regular DQN approach is of limited applicability for yielding an acceptable approximation due to the complexity incurred by $\{h_n\}$. When a DQN is used, the optimal Q-value function is approximated by an ANN with low-dimensional representations. It is of paramount importance that resultant representation has strong generalization capability such that the agent can make sensible decisions in states that are unprecedented or rarely visited. Typically, the more complex the approximation function is, the poorer generalization the trained DQN can achieve. Including $\{h_n\}$ as part of the inputs to the ANN turns out to be troublesome since it significantly increases the dimensionality of the function mapping. In addition, the dynamics of $\{h_n\}$ imply that the scale of rewards varies greatly. Conventionally, the DQN approach clips all the positive rewards to be 1 to stabilize the training process. However, ignoring the information of $\{h_n\}$ does not make sense for the scheduling problem since it will lead to a policy that always tries to perform the easiest updates. Those challenges suggest a separate consideration of $\{h_n\}$ rather than feeding them into the ANN. Observing that the reward function defined in Section IV-A is additive, we decompose $Q^{\mu}(s, o)$ via reward decomposition such that several simpler option-value functions need to be estimated. Specifically, the option-value function for a policy μ can be decomposed as follows:

$$\begin{aligned} Q^{\mu}(s, o) &= \mathbb{E}\left[\sum_{\tau=t}^{T} R(\tau) \middle| \mathcal{E}(o \circ \mu, s, t)\right] \\ &= \sum_{n=1}^{N} \mathbb{E}\left[\sum_{\tau=t}^{T} r_n(\tau) \middle| \mathcal{E}(o \circ \mu, s, t)\right] = \sum_{n=1}^{N} q_n^{\mu}(s, o) \end{aligned} \tag{20}$$

where $q_n^{\mu}(s, o)$ is the option-value functions for the policy μ under reward signal r_n.

Decomposing the option-value function $Q^{\mu}(s, o)$ in the applicationwise manner facilitates the exploitation of domain knowledge. Since fine-grained reward signal is utilized, we can circumvent the reward clipping technique by estimating the surrogate function $\hat{q}_n^{\mu}(s, o)$, with the reward signal

$$\hat{r}_n = \begin{cases} d_n', & \text{if } d_n = 0 \\ 0, & \text{otherwise.} \end{cases} \tag{21}$$

Since the information of $\{h_n\}$ is known at the beginning of each frame, $q_n^{\mu}(s, o)$ can be directly evaluated with the estimation of $\hat{q}_n^{\mu}(s, o)$. In this way, $Q^{\mu}(s, o)$ is estimated via the surrogate rewards $\{\hat{r}_n\}$, where $\hat{r}_n \in \{0, 1\}$. Moreover, since no further reward with respect to application n can be received if $\beta_n(s) = 1$, we can safely set $q_n^{\mu}(s, o) = 0$ if terminal states are encountered during the training process. Therefore, the parameters $\boldsymbol{\theta}$ in the DQN can be fully used to represent $q_n^{\mu}(s, o)$ in the case of $\beta_n(s) = 0$.

Consider that $Q^*(s, o)$ is an aggregation of all $q_n^*(s, o)$, the optimal next option o' is determined by the estimations of all $q_n^*(s, o)$. In other words, $q_n^*(s, o)$ loses the recursive property the Bellman optimality equation of $Q^*(s, o)$ enjoys. Similar to the HRA approach [24], we train the DQN by evaluating a stationary random policy ρ, where an option is selected uniformly at random from the set of nonterminal options, say $\mathcal{O}_s$. Under this policy, the Bellman operator of $Q^{\mu}(s, o)$ is decomposable. Therefore, $\boldsymbol{\theta}$ can be updated by using the expected

Sarsa method [30], with the target value

$$y_n^o = \hat{r}_n + \hat{u}_n(s', o) \tag{22}$$

where

$$\hat{u}_n(s,o) = (1-\beta_o(s))\tilde{q}_n\big(s,o;\boldsymbol{\theta}^-\big) + \frac{\beta_o(s)}{|\mathcal{O}_s|}\sum_{o'\in\mathcal{O}_s}\tilde{q}_n\big(s,o';\boldsymbol{\theta}^-\big). \tag{23}$$

The corresponding loss function is written as

$$\mathcal{L}(\boldsymbol{\theta}) = \mathbb{E}_{(s,o,\mathbf{r},s')}\left[\sum_{\hat{o}\in\Xi_s^o}\sum_{n=1}^{N}\Big(y_n^{\hat{o}} - \hat{q}\big(s,\hat{o};\boldsymbol{\theta}\big)\Big)^2\right] \tag{24}$$

where Ξ_s^o represents the set of options that will schedule the same device as option o does.

Another benefit of using the stationary random policy ρ is that the complexity of approximating $\hat{q}_n^\rho(s,o)$ can be further amortized by excluding $\{h_n\}$ from the inputs of the ANN. Under policy ρ, the reward signal $\hat{r}_n$ is independent of $\{h_n\}$. Therefore, the information of $\{h_n\}$ is irrelevant to the update of $\hat{q}_n(s,o;\boldsymbol{\theta})$. According to [24], such irrelevant information only adds noise to the learning process. Let $\hat{s} \triangleq [u_1, u_2, \ldots, u_M; t]$. Since only the information of $\hat{s}$ is necessary in approximating $\hat{q}_n^\rho(s,o)$, we rewrite the parameterized representation of the ANN for $\hat{q}_n^\rho(s,o)$ as $\hat{q}_n(\hat{s},o;\boldsymbol{\theta})$, indicating that the irrelevant information $\{h_n\}$ is not used by the ANN.[3] In this way, the semiconsistent approximation to $Q^*(s,o)$ is represented as

$$\hat{Q}(s,o;\boldsymbol{\theta}) = \sum_{n=1}^{N}\hat{q}_n\big(\hat{s},o;\boldsymbol{\theta}\big) * h_n(h_n+2). \tag{25}$$

Instead of performing the nonlinear transformations via an ANN, information of $\{h_n\}$ is utilized explicitly in the above expression by leveraging the additive structure of the scheduling problem. Compared to $Q^*(s,o)$, $\hat{q}_n^\rho(\hat{s},o)$ is easier to learn since it characterizes a clearer connection between the option selections and the outcomes.

B. Attention-Integrated Relevance Network Architecture

The $\hat{q}$-value function $\hat{q}_n^\rho(\hat{s},o)$ characterizes the probability that the status of application n can be updated while the agent executes the option o. Intuitively, executing different options may have a similar impact on the status update of a certain application. For example, consider a set of IoT devices $(1,2,3,4,5)$, each application can be represented by a subset of the device set. With error-free transmissions assumed, the effect of executing the option of application $(1,4)$ on the status update of application $(1,2,3)$ is almost the same as that of executing the option of application $(2,5)$. Likewise, the option values of executing a specific option with respect to different applications may be close to each other. In this sense, values of $\hat{q}_n^\rho(\hat{s},o)$ for different combinations of application n and option o are highly correlated. Taking this into account, we embrace the idea of DRRN [25] to design the network architecture of the DQN such that the correlations among applications can be modeled in an expressive fashion. That is, $\hat{q}_n(\hat{s},o;\boldsymbol{\theta})$ is

[3]The value of the termination condition $\beta_n(s)$ does not rely on h_n, either.

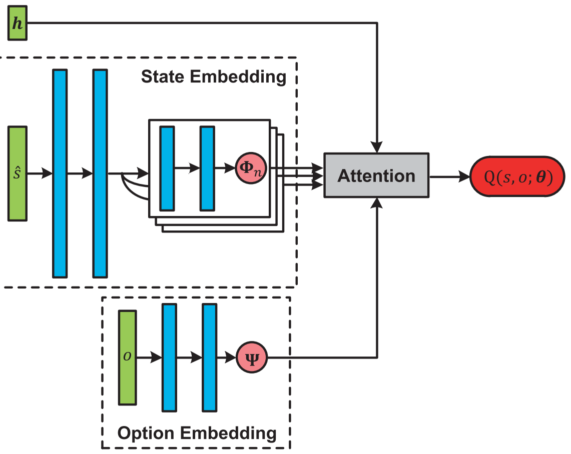


Fig. 3. ARN (the blue blocks represent the hidden layers that consist of trainable parameters).

expressed as a pairwise interaction between a state embedding vector for application n and the embedding vector for option o.

As illustrated in Fig. 3, we build two separate ANNs to embed the state space and option space, respectively. Formally, let $\{\boldsymbol{\Phi}_n(\hat{s},\boldsymbol{\theta}_s)\}$ be the outputs of the former ANN and $\boldsymbol{\Psi}(o,\boldsymbol{\theta}_o)$ denote the embedding vector of option o generated by the later ANN. Given the scale of the estimations, the scaled dot product is used as the interaction function. Specifically, $\hat{q}_n^\rho(\hat{s},o)$ is estimated as

$$\hat{q}_n\big(\hat{s},o;\boldsymbol{\theta}\big) = \frac{\boldsymbol{\Phi}_n^\top\big(\hat{s},\boldsymbol{\theta}_s\big)\boldsymbol{\Psi}(o,\boldsymbol{\theta}_o)}{\sqrt{V}} \tag{26}$$

where $(\cdot)^\top$ represents the transpose operator and V is the dimension of each embedding vector, e.g., $\boldsymbol{\Phi}_n(\hat{s},\boldsymbol{\theta}_s)$ or $\boldsymbol{\Psi}(o,\boldsymbol{\theta}_o)$. Combining (25) and (26), $\hat{Q}(s,o;\boldsymbol{\theta})$ can be considered as the outcome of a simplified scaled dot-product attention mechanism [31] with inputs $\{\boldsymbol{\Phi}_n\}$, $\boldsymbol{\Psi}$, and $\{h_n\}$. In this sense, the proposed DQN is called ARN. The state embedding network has a multihead structure, where each head outputs the state embedding vector for a particular application. The collection of heads can share several lower level layers of the ANN.

The training process follows a practice common for training a DQN, which is presented in Algorithm 1. The He initialization method [32] and Glorot initialization method [33] are used to initialize the network weights in the hidden layers and output layer, respectively (line 1). Given that the experience samples of idle slots are useless for the learning agent, those samples will not be added into the replay buffer (lines 9 and 10). We apply the ϵ-greedy policy to balance the exploration–exploitation tradeoff. That is, a random option will be selected with probability ϵ; otherwise, the option that yields the maximum value of $Q(s,o,\boldsymbol{\theta})$ is chosen (line 13). The value of ϵ decays from 1 to 0.05 during the training process. Since an experienced sample may be used to update multiple $\hat{q}$-values, the popular experience sampling method—prioritized experience replay (PER) [34]—cannot be applied directly. In this article, we sample a minibatch of transitions via the combined experience replay (CER) method (line 19), where the

Algorithm 1 DQN Training

```
1: Initialize $\{\mathbf{\Phi}_n(\hat{s}, \boldsymbol{\theta}_s)\}$ and $\mathbf{\Psi}(o, \boldsymbol{\theta}_o)$ with random weights $\boldsymbol{\theta}_s$ and $\boldsymbol{\theta}_o$
2: Initialize the target networks $\{\tilde{\mathbf{\Phi}}_n(\hat{s}, \boldsymbol{\theta}_s^-)\}$ and $\tilde{\mathbf{\Psi}}(o, \boldsymbol{\theta}_o^-)$ with $\boldsymbol{\theta}_s^- = \boldsymbol{\theta}_s$ and $\boldsymbol{\theta}_o^- = \boldsymbol{\theta}_o$
3: Initialize the replay buffers $\mathcal{B}$
4: for $k = 1, \ldots, K$ do
5:     Update $\{h_n\}$
6:     Term $\leftarrow 0$, $o \leftarrow 0$
7:     for $t = 1, \ldots, T$ do
8:         Update $s$
9:         if $\bigwedge_{n=1}^{N} \beta_n(\hat{s}_n) = 1$ then
10:            Keep idle
11:        else
12:            if Term = 0 then
13:                Select option $o$ according to the $\epsilon$-greedy policy
14:            end if
15:            Schedule a device according to option $o$'s policy; Receive rewards $\{\hat{r}_n\}$ and observe $s'$
16:            Add transition $(s, o, \mathbf{r}, s')$ into $\mathcal{B}$
17:            Term $\leftarrow \beta_o(s')$
18:        end if
19:        Sample a random minibatch of transitions from $\mathcal{B}$
20:        Perform a gradient descent step on $\boldsymbol{\theta}_s$ and $\boldsymbol{\theta}_o$ with respect to the loss function (24)
21:        Update the target networks:
           $\boldsymbol{\theta}_s^- \leftarrow \alpha\boldsymbol{\theta}_s + (1-\alpha)\boldsymbol{\theta}_s^-$
           $\boldsymbol{\theta}_o^- \leftarrow \alpha\boldsymbol{\theta}_o + (1-\alpha)\boldsymbol{\theta}_o^-$
22:    end for
23: end for
```

latest sample has the highest priority [35]. Instead of updating the target DQN periodically, the method of "soft" target update [36] is used to update the weights in the target networks (line 21). With $\alpha \ll 1$, the target networks track the learned networks quite slowly.

Remark: Although Algorithm 1 is presented in an online setting, the training process can be performed offline given that the expected SARSA method is used in an off-policy manner. In this way, the learning agent can derive the policy out of historical data, e.g., a set of transitions generated by other policies. To completely transform the proposed algorithm to a batch RL algorithm, however, extra considerations are necessary, e.g., how to efficiently perform the importance sampling. We leave it as our future work.

VI. Performance Evaluation

In this section, numerical results are presented to evaluate the performance of the DRL-based approach. We consider IoT systems with 50 devices. The number of devices that are associated with an application is randomly selected from $\{2, 3, 4, 5\}$. Therefore, depending on the number of applications, the actual number of devices which are involved in the scheduling problem varies between 19 and 41. The probabilities of successful transmission p_m follow a uniform distribution, ranging from 0.3 to 1. Unless otherwise stated, the length of each frame is set to 10. The ANNs in the ARN model are implemented using TensorFlow [37]. The state embedding network consists of a shared stack and multiple heads. The shared stack contains two hidden layers, between which is a batch normalization layer. Each hidden layer has 200 units with ReLU activation. We adopt the method of dueling network architecture [38] to further smooth out the learning such that the advantage function rather than the $\hat{q}$-value function is approximated via the scaled dot-product operation. The output of the shared stack is routed to the head blocks, each of which consists of 100 ReLU units followed by $|V|+1$ units with tanh activation. The option embedding network has the same architecture as the aforementioned shared stack except that 100 ReLU units are used in each hidden layer. The output of the second hidden layer is fed to the output layer with V tanh units. We choose the tanh function because it is zero centered and has bounded output value within the range -1 to 1. More of the detailed hyperparameter settings can be found in Table I.

TABLE I
HYPERPARAMETERS SETTINGS

Hyperparameter	Value
minibatch size	32
replay buffer size	1000
learning rate	0.00025
target update coefficient α	0.01
number of episodes K	50000
dimension of embedding vectors V	15

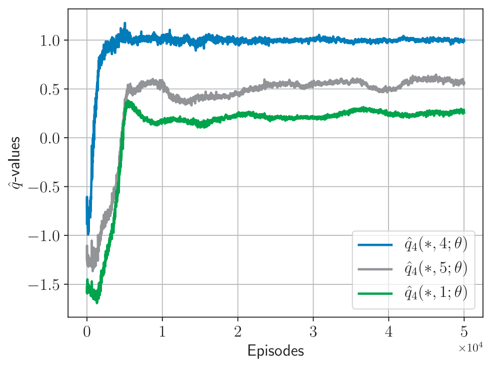


Fig. 4. Training curves tracking $\hat{q}$-values of representative (n, o) pairs.

A. Semiconsistent Approximation

Before discussing the performance achieved by the DRL-based approach, we use some representative examples to demonstrate that our approach robustly learns a reasonable policy that captures the correlations across the applications. Here, we consider that there are total 19 devices that are associated with ten applications, say $M = 19$ and $N = 10$. The training curves, which track the predicted $\hat{q}$-values of three (n, o) pairs at the beginning of a frame,[4] are illustrated in Fig. 4. Recall that each application can be represented by a set

[4]The state at this moment is denoted by $*$.

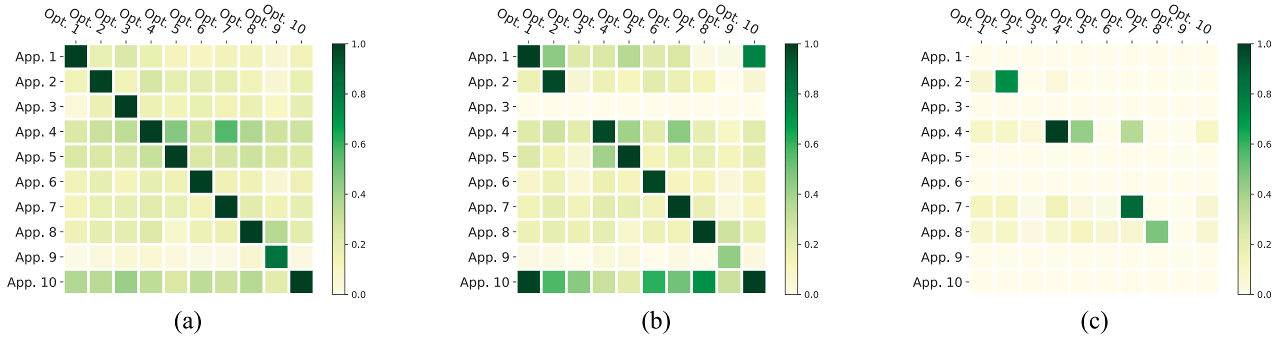


Fig. 5. Snapshots of $\hat{q}$-values. (a) $t = 1$. (b) $t = 5$. (c) $t = 9$.

TABLE II
COMPARISONS OF THE AoCI PERFORMANCES

Settings	I: $N = 10, M = 19$			II: $N = 15, M = 32$			III: $N = 20, M = 41$		
	Average	90th %ile	99th %ile	Average	90th %ile	99th %ile	Average	90th %ile	99th %ile
ARN	**26.48**	**49**	**67**	**61.28**	**119**	**190**	**71.90**	**136**	**187**
GREEDY	29.20	55	70	72.61	136	193	83.04	152	198
RAND	45.71	103	207	120.72	280	785	144.83	338	793

of devices, the applications involved in Fig. 4 are App. 1:(1, 2, 3, 4, 11), App. 4:(7, 8), and App. 5:(3, 7, 9).

From Fig. 4, we can observe that each predicted $\hat{q}$-value reaches a relatively stable value. Specifically, $\hat{q}_4(*, 1; \boldsymbol{\theta}) \approx 0.27$, $\hat{q}_4(*, 4; \boldsymbol{\theta}) \approx 1$, and $\hat{q}_4(*, 5; \boldsymbol{\theta}) \approx 0.55$. Given that App. 4 relies on the information from two devices, it is almost certain that the status of this application can be updated by the end of a frame if option 4 is selected at the beginning of that frame. However, if option 1 is executed, the status of App. 4 will be updated with a low probability since App. 1 requires the information from four devices which are all irrelevant to App. 4. Compared to App. 1, App. 5 needs information from a smaller number of devices. Moreover, one of these devices is also associated with App. 4. Therefore, it also makes sense that executing option 5 leads to a higher $\hat{q}$-value with respect to App. 4.

With the trained model, snapshots of $\hat{q}$-values of all (n, o) pairs in an episode are presented in Fig. 5. We choose three cases: 1) $t = 1$; 2) $t = 5$; and 3) $t = 9$, which, respectively, represent the moments in the beginning, in the middle, and near the end of a frame. As shown in Fig. 5(a), for each application, executing its corresponding option always results in the largest $\hat{q}$-value. Most of those values in the diagonal direction are approximately equal to 1 at the beginning of the frame. The relative magnitude of values in the same row indicates the relevances of options to a particular application. App. 9 is an application that does not share any devices with other applications and two of the devices it relies on have poor channels. Thus, the status of this application can only be updated by executing option 9. In the case of Fig. 5(b), option 3 is completed and option 1 is being executed. We can observe that the $\hat{q}$-value of executing option 9 with respect to App. 9 decreases since there are fewer slots left. Another interesting observation is that the $\hat{q}$-value of executing option 1 with respect to App. 10 becomes almost 1. This is because the set of devices for App. 10 is a subset of the union of the device sets for App. 3 and App. 1. When $t = 9$, there are only two slots left. As shown in Fig. 5(c), applications with nonterminal corresponding option benefit mainly from executing their options. Executing option 5 or 7 yields a nonnegligible $\hat{q}$-value with respect to App. 4. This is because executing any one of the two options will lead to the same result as executing option 4, i.e., scheduling the same device at the current slot.

B. Performance Comparisons

We compare the DRL-based approach, which is denoted by ARN, with the following two baseline methods.

1) *AoCI Greedy Policy (GREEDY):* Devices that are associated with the application that has the largest AoCI will be scheduled in sequence first.
2) *Randomized Policy (RAND):* Application is selected uniformly at random from the set of nonterminal options. The devices which are associated with the selected application will be scheduled in sequence.

These approaches are evaluated over different system settings. Specifically, we consider systems with 10, 15, and 20 applications, which involves 19, 32, and 41 devices, respectively. The long-term average and some other useful statistics of the AoCI performance are summarized in Table II. In terms of the long-term average AoCI, the DRL-based approach outperforms the other two methods in all three settings. Recall that the RAND policy is used as the target policy in the training phase, the significant performance boost achieved by the ARN policy indicates that the learning agent is able to distill insightful knowledge and obtain a semiconsistent approximation to the option-value function. Compared to the GREEDY policy, the proposed approach achieves 9%, 16%, and 13% improvements in those three settings, respectively. The 90th percentile and 95th percentile of the values of AoCI under

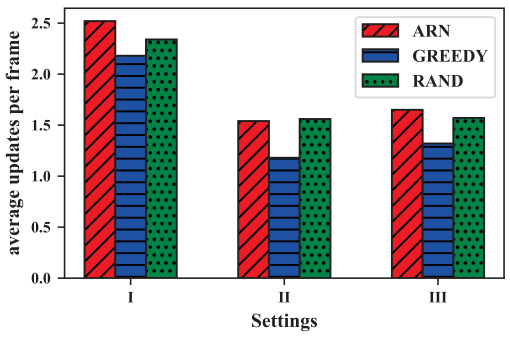


Fig. 6. Average number of applications which are updated at each frame.

different policies are also presented. As shown in Table II, values of AoCI generated by the ARN policy have better tail behavior than the other two methods.

Furthermore, we investigate the average number of applications these three policies can update at each frame, which is illustrated in Fig. 6. It turns out that the GREEDY policy always achieves the least updates, even less than that fulfilled by the RAND policy. This is because the preference of the GREEDY policy to the application with the largest AoCI makes it pay more attention, compared to the RAND policy, to those hard-to-update applications, which typically requires relatively more transmissions. Given the poor AoCI performance of the RAND policy, a certain portion of updates achieved by the RAND policy, e.g., updates of the applications which have low AoCI, are in some sense ineffective in improving the average AoCI. In other words, such a preference is meaningful in maintaining a good AoCI performance. Nevertheless, naively sticking to the application with the largest AoCI may waste a portion of slots. For example, consider the case that the application with the largest AoCI still requires the information from four devices while the current frame has only five slots left. Definitely, the GREEDY policy will execute the option that is associated with this application, which probably fails to finish by the end of the current frame and will be reexecuted at the beginning of the next frame. Taking the application correlations and the probabilities of the update completion into account, the ARN policy may execute other options in the aforementioned case to increase the number of updates, as exemplified in Fig. 5(b). Moreover, the option chosen by the GREEDY policy will also be executed by the ARN policy at the beginning of the next frame, as indicated in Fig. 5(a). In this way, the ARN policy can effectively plan the transmission schedule to improve the system freshness by jointly considering the quality and the quantity of the updates.

C. Impact of the Frame Length

Fig. 7 presents the time-average AoCI under the ARN policy with different frame lengths. As shown in Fig. 7, given a system setting, there is a sweet spot in terms of the frame length T, where the lowest time-average AoCI is achieved. With a small T, only a small portion of applications can be updated at each frame, which yields high time-average AoCI. On the other hand, when T is relatively large, although more messages can be transmitted during a frame, the waiting time of status update becomes longer since the controller updates the applications at the end of each frame. In addition, it is illustrated in Fig. 7 that the ARN can offer superior performance over the GREEDY policy. The reasons that the advantage of the ARN policy diminishes as the frame length increases are twofold. The first one is the generalization degradation. Recall that the learning agents used by the ARN policy are trained with $T = 10$ when tested in a setting with different frame length, the distribution of the state changes. Particularly, many system states are not even in the state space of the original problem, e.g., states with $t > 10$. Such an issue is commonly mitigated by fine tuning the models based on the system setting, i.e., the exact frame length that is used in the system. The other one is due to the nature of the scheduling problem. That is, the optimality gap of the GREEDY policy decreases with an increase of T. This phenomenon can be observed clearly in Fig. 7(a). Note that there are 19 devices involved in this setting. When $T = 25$, it is probably the GREEDY policy itself is near optimal and thus marginal performance boost can be achieved by the ARN policy. Nevertheless, we argue that in practical systems, the setting that the value of frame length is larger than the number of devices is more reasonable and common, where the proposed approach can be applied to improve the system's information freshness.

VII. Related Work

As an emerging topic in the area of wireless networking, AoI-based link scheduling optimization has been explored via different approaches. The work in [9] demonstrated that the problem of wireless link scheduling with age minimization is NP-hard in general. Consider the NP-hardness of the scheduling problem, a couple of heuristics are proposed in [12]–[14], among which the greedy policy is shown to be optimal in symmetric network settings [13] and the round-robin policy was proven to be asymptotically optimal [12]. Observing that the dynamics of AoI can be characterized by a virtual queue, the works in [10], [13], [14], and [17] developed scheduling policies by leveraging the Lyapunov optimization theory. With MDP formulations, analytical results for scheduling problems in different scenarios were presented in [11] and [18]. The scheduling problems were modeled as restless multiarmed bandit (RMAB) problems in [13]–[16] and [18], where Whittle's index policies were, respectively, derived. The work in [39] studied the AoI minimization problem with nonuniform status packet sizes and used the MDP methods to design a joint scheduling and sampling policy while a unified sampling and scheduling approach is proposed in [40] to minimize the tracking errors of status updates. A common limitation of these approaches is that they apply for weakly coupled systems in which the relationships between the sources and the information of interest are one to one.

The work in [19] pointed out the potential correlations across multiple information flows in wireless camera networks, where images from multiple cameras are processed jointly for 3-D scene reconstruction. With the objective of minimizing the AoCI, a greedy strategy was proposed for the transmission schedule. In this article, we generalize the AoCI

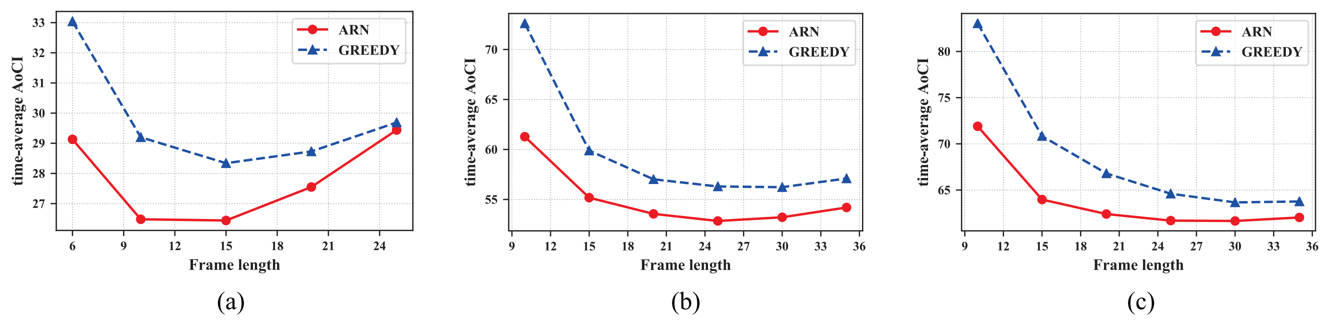


Fig. 7. Time-average AoCI versus different frame length. (a) $N = 10$ and $M = 19$. (b) $N = 15$ and $M = 32$. (c) $N = 20$ and $M = 41$.

model proposed in [19] such that many-to-many associations between devices and applications are considered. In many wireless networking problems, temporal dependency has a dramatic impact on the system performance and has been exploited through various approaches [41]–[43]. This article leverages the emerging DRL techniques to model the correlations across multiple information sources and exploit the temporally extended action.

Recently, there are few works that also investigated the applicability of RL/DRL methods to develop control algorithms that optimize AoI in different networking context [44]–[47]. It was shown in these works that direct applications of general learning frameworks, e.g., Q-learning, DQN approach, or actor–critic algorithm can provide a significant performance gain in terms of AoI. Due to the correlations that arise in the scheduling problem, however, the global Q-value function is so complicated that applying the DQN approach directly cannot lead to a stable approximation. Therefore, this article tailors the general learning framework to the scheduling problem such that domain knowledge can be exploited in various ways to learn a semiconsistent approximation, which yields a reasonable scheduling policy.

VIII. Conclusion

In this article, we have investigated the uplink transmission scheduling problem for IoT systems in which the application-level services rely on the timely delivery of information from multiple devices. By formulating the scheduling problem as an episodic MDP problem with the application-oriented policy, we have developed a DRL-based approach to optimize the long-term average AoCI, which adopts the reward decomposition technique for complexity reduction and introduces an attention-based relevance network architecture that captures the correlations among applications. Through extensive simulations, we have shown that the proposed approach can robustly learn a semiconsistent approximation to the option-value function, yielding a scheduling policy that outperforms some conventional algorithms.

References

[1] S. Kaul, M. Gruteser, V. Rai, and J. Kenney, “Minimizing age of information in vehicular networks,” in *Proc. IEEE Annu. Commun. Soc. Conf. Sensor Mesh Ad Hoc Commun. Netw. (SECON)*, 2011, pp. 350–358.

[2] A. Kosta, N. Pappas, and V. Angelakis, “Age of information: A new concept, metric, and tool,” *Found. Trends Netw.*, vol. 12, no. 3, pp. 162–259, 2017.

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

[4] R. D. Yates, “Lazy is timely: Status updates by an energy harvesting source,” in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, 2015, pp. 3008–3012.

[5] 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, Nov. 2017.

[6] M. Costa, M. Codreanu, and A. Ephremides, “Age of information with packet management,” in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, 2014, pp. 1583–1587.

[7] A. M. Bedewy, Y. Sun, and N. B. Shroff, “Optimizing data freshness, throughput, and delay in multi-server information-update systems,” in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, 2016, pp. 2569–2573.

[8] Y. Sun, E. Uysal-Biyikoglu, and S. Kompella, “Age-optimal updates of multiple information flows,” in *Proc. IEEE INFOCOM Workshops*, 2018, pp. 136–141.

[9] Q. He, D. Yuan, and A. Ephremides, “Optimal link scheduling for age minimization in wireless systems,” *IEEE Trans. Inf. Theory*, vol. 64, no. 7, pp. 5381–5394, Jul. 2018.

[10] C. Joo and A. Eryilmaz, “Wireless scheduling for information freshness and synchrony: Drift-based design and heavy-traffic analysis,” in *Proc. IEEE WiOpt*, 2017, pp. 1–8.

[11] Y.-P. Hsu, E. Modiano, and L. Duan, “Age of information: Design and analysis of optimal scheduling algorithms,” in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, 2017, pp. 561–565.

[12] Z. Jiang, B. Krishnamachari, X. Zheng, S. Zhou, and Z. Niu, “Timely status update in wireless uplinks: Analytical solutions with asymptotic optimality,” *IEEE Internet Things J.*, vol. 6, no. 2, pp. 3885–3898, Apr. 2019.

[13] I. Kadota, A. Sinha, E. Uysal-Biyikoglu, R. Singh, and E. Modiano, “Scheduling policies for minimizing age of information in broadcast wireless networks,” *IEEE/ACM Trans. Netw.*, vol. 26, no. 6, pp. 2637–2650, Jun. 2018.

[14] I. Kadota, A. Sinha, and E. Modiano, “Optimizing age of information in wireless networks with throughput constraints,” in *Proc. IEEE INFOCOM*, 2018, pp. 1844–1852.

[15] Y.-P. Hsu, “Age of information: Whittle index for scheduling stochastic arrivals,” in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, 2018, pp. 2634–2638.

[16] Z. Jiang, B. Krishnamachari, S. Zhou, and Z. Niu, “Can decentralized status update achieve universally near-optimal age-of-information in wireless multiaccess channels?” in *Proc. IEEE Int. Test Conf. (ITC)*, vol. 1, 2018, pp. 144–152.

[17] N. Lu, B. Ji, and B. Li, “Age-based scheduling: Improving data freshness for wireless real-time traffic,” in *Proc. ACM MobiHoc*, 2018, pp. 191–200.

[18] B. Yin *et al.*, “Only those requested count: Proactive scheduling policies for minimizing effective age-of-information,” in *Proc. IEEE INFOCOM*, 2019, pp. 109–117.

[19] Q. He, G. Dan, and V. Fodor, “Minimizing age of correlated information for wireless camera networks,” in *Proc. IEEE INFOCOM Workshops*, 2018, pp. 547–552.

[20] M. Chowdhury and I. Stoica, “CoFlow: A networking abstraction for cluster applications,” in *Proc. ACM HotNets*, 2012, pp. 31–36.

[21] C. J. Watkins and P. Dayan, "*Q*-learning," *Mach. Learn.*, vol. 8, nos. 3–4, pp. 279–292, 1992.

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

[23] R. S. Sutton, D. Precup, and S. Singh, "Between MDPS and semi-MDPS: A framework for temporal abstraction in reinforcement learning," *Artif. Intell.*, vol. 112, nos. 1–2, pp. 181–211, 1999.

[24] H. Van Seijen, M. Fatemi, J. Romoff, R. Laroche, T. Barnes, and J. Tsang, "Hybrid reward architecture for reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5392–5402.

[25] J. He *et al.*, "Deep reinforcement learning with a natural language action space," 2015. [Online]. Available: arXiv:1511.04636.

[26] L. Tassiulas and A. Ephremides, "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks," in *Proc. IEEE CDC*, 1990, pp. 2130–2132.

[27] M. J. Neely, "Stochastic network optimization with application to communication and queueing systems," *Synth. Lectures Commun. Netw.*, vol. 3, no. 1, pp. 1–211, 2010.

[28] Y. Cheng, H. Li, D. M. Shila, and X. Cao, "A systematic study of maximal scheduling algorithms in multiradio multichannel wireless networks," *IEEE/ACM Trans. Netw.*, vol. 23, no. 4, pp. 1342–1355, Aug. 2015.

[29] U. Feige, D. Peleg, and G. Kortsarz, "The dense *k*-subgraph problem," *Algorithmica*, vol. 29, no. 3, pp. 410–421, 2001.

[30] H. Van Seijen, H. Van Hasselt, S. Whiteson, and M. Wiering, "A theoretical and empirical analysis of expected SARSA," in *Proc. IEEE ADPRL*, 2009, pp. 177–184.

[31] A. Vaswani *et al.*, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5998–6008.

[32] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, 2015, pp. 1026–1034.

[33] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. AISTATS*, 2010, pp. 249–256.

[34] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2016, pp. 449–458.

[35] S. Zhang and R. S. Sutton, "A deeper look at experience replay," 2017. [Online]. Available: arXiv:1712.01275.

[36] T. P. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2016, pp. 1–142.

[37] M. Abadi *et al.* (2015). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. [Online]. Available: http://tensorflow.org/

[38] Z. Wang, T. Schaul, M. Hessel, H. Hasselt, M. Lanctot, and N. Freitas, "Dueling network architectures for deep reinforcement learning," in *Proc. ICML*, 2016, pp. 1995–2003.

[39] B. Zhou and W. Saad, "Minimizing age of information in the Internet of Things with non-uniform status packet sizes," in *Proc. IEEE ICC*, 2019, pp. 1–6.

[40] Z. Jiang, S. Zhou, Z. Niu, and C. Yu, "A unified sampling and scheduling approach for status update in multiaccess wireless networks," in *Proc. IEEE INFOCOM*, 2019, pp. 208–216.

[41] H. Zhu, S. Chang, M. Li, K. Naik, and S. Shen, "Exploiting temporal dependency for opportunistic forwarding in urban vehicular networks," in *Proc. IEEE INFOCOM*, 2011, pp. 2192–2200.

[42] H. Zhu, M. Dong, S. Chang, Y. Zhu, M. Li, and X. S. Shen, "ZOOM: Scaling the mobility for fast opportunistic forwarding in vehicular networks," in *Proc. IEEE INFOCOM*, 2013, pp. 2832–2840.

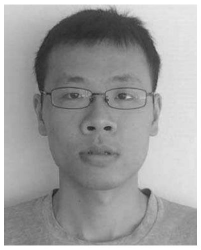
[43] J. Qin, H. Zhu, Y. Zhu, L. Lu, G. Xue, and M. Li, "POST: Exploiting dynamic sociality for mobile advertising in vehicular networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 6, pp. 1770–1782, Jun. 2015.

[44] E. T. Ceran, D. Gündüz, and A. György, "A reinforcement learning approach to age of information in multi-user networks," in *Proc. IEEE Int. Symp. Pers. Indoor Mobile Radio Commun. (PIMRC)*, 2018, pp. 1967–1971.

[45] E. Sert, C. Sönmez, S. Baghaee, and E. Uysal-Biyikoglu, "Optimizing age of information on real-life TCP/IP connections through reinforcement learning," in *Proc. IEEE Signal Process. Commun. Appl. Conf. (SIU)*, 2018, pp. 1–4.

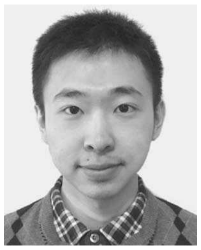
[46] A. Elgabli, H. Khan, M. Krouka, and M. Bennis, "Reinforcement learning based scheduling algorithm for optimizing age of information in ultra reliable low latency networks," in *Proc. IEEE Int. Symp. Comput. Commun. (ISCC)*, 2019, pp. 1–6.

[47] M. A. Abd-Elmagid, A. Ferdowsi, H. S. Dhillon, and W. Saad, "Deep reinforcement learning for minimizing age-of-information in UAV-assisted networks," 2019. [Online]. Available: arXiv:1905.02993.



Bo Yin (Student Member, IEEE) received the B.E. degree in electronic information engineering and the M.E. degree in electronic science and technology from Beihang University, Beijing, China, in 2010 and 2013, respectively. He is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering, Illinois Institute of Technology, Chicago, IL, USA.

His research interests include network security, network resource allocation, and ML-based network optimization.



Shuai Zhang (Student Member, IEEE) received the B.Eng. degree from Zhejiang University, Hangzhou, China, in 2013, and the M.S. degree from the University of California at Los Angeles, Los Angeles, CA, USA, in 2015. He is currently pursuing the Ph.D. degree with Illinois Institute of Technology, Chicago, IL, USA.

His research interests include wireless communication and distributed learning.

Yu Cheng (Senior Member, IEEE) received the B.E. and M.E. degrees in electronic engineering from Tsinghua University, Beijing, China, in 1995 and 1998, respectively, and the Ph.D. degree in electrical and computer engineering from the University of Waterloo, Waterloo, ON, Canada, in 2003.

He is currently a Full Professor with the Department of Electrical and Computer Engineering, Illinois Institute of Technology, Chicago, IL, USA. His research interests include wireless network performance analysis, network security, big data, cloud computing, and machine learning.

Prof. Cheng received the Best Paper Award at QShine 2007 and IEEE ICC 2011, the Runner-Up Best Paper Award at ACM MobiHoc 2014, the National Science Foundation CAREER Award in 2011, and the IIT Sigma Xi Research Award in the Junior Faculty Division in 2013. He is an Associate Editor for the IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, the IEEE INTERNET OF THINGS JOURNAL, and IEEE WIRELESS COMMUNICATIONS. He has served as the Symposium Co-Chair for IEEE ICC and IEEE GLOBECOM, and the Technical Program Committee Co-Chair for WASA 2011 and ICNC 2015. He was a Founding Vice Chair of the IEEE ComSoc Technical Subcommittee on Green Communications and Computing. He was an IEEE ComSoc Distinguished Lecturer from 2016 to 2017.