Caching Transient Content for IoT Sensing: Multi-Agent Soft Actor-Critic

Xiongwei Wu, *Student Member, IEEE*, Xiuhua Li, *Member, IEEE*, Jun Li, *Senior Member, IEEE*, P. C. Ching, *Life Fellow, IEEE*, Victor C. M. Leung, *Life Fellow, IEEE*, H. Vincent Poor, *Life Fellow, IEEE*

***Abstract*—Edge nodes (ENs) in Internet of Things commonly serve as gateways to cache sensing data while providing accessing services for data consumers. This paper considers multiple ENs that cache sensing data under the coordination of the cloud. Particularly, each EN can fetch content generated by sensors within its coverage, which can be uploaded to the cloud via fronthaul and then be delivered to other ENs beyond the communication range. However, sensing data are usually transient with time whereas frequent cache updates could lead to considerable energy consumption at sensors and fronthaul traffic loads. Therefore, we adopt Age of Information to evaluate data freshness and investigate intelligent caching policies to preserve data freshness while reducing cache update costs. Specifically, we model the cache update problem as a cooperative multi-agent Markov decision process with the goal of minimizing the long-term average weighted cost. To efficiently handle the exponentially large number of actions, we devise a novel reinforcement learning approach, which is a discrete multi-agent variant of soft actor-critic (SAC). Furthermore, we generalize the proposed approach into a decentralized control, where each EN can make decisions based on local observations only. Simulation results demonstrate the superior performance of the proposed SAC-based caching schemes.**

***Index Terms*—Internet of Things, age of information, cooperative multi-agent Markov decision process, soft actor-critic**

I. Introduction

With the advancement of wireless access technology, it is envisioned that hundreds of billions of devices will access the Internet, forming the so called *Internet of Things (IoT)* [1]. The advent of this paradigm generalizes the accessibility towards various kinds of IoT sensors (e.g., smart cameras and temperature sensors), and thus enables intelligent services to improve human quality of life [2], [3]. However, countless electronic devices are anticipated to generate a large volume of traffic loads, which can possibly make networks saturate and degrade the quality of service. To overcome these challenges, edge nodes (ENs), e.g., small-cell base stations, are expected to act as gateways to cache sensing data close to the consumers. Consequently, it can greatly reduce traffic loads, transmission delay, and energy cost in IoT sensing networks [4], [5].

Currently, some existing studies have been devoted to caching policies at wireless networks in terms of optimizing communication performance criteria, e.g., traffic loads, latency, and power consumption [6]–[11]. These caching policies emphasize how to efficiently cache multimedia content given limited storage at ENs. However, IoT sensors usually generate sensing data at a relatively small size [12]. Therefore, each EN can be assumed to have enough storage to cache content items produced by all sensors in the network [12]. In this way, each EN can locally satisfy user requests towards all content items. Moreover, in contrast with multimedia content that is often *in-transient*, sensing data cached at ENs gradually become outdated as time passes. The staleness of caching content may significantly deteriorate the performance of IoT sensing services. Indeed, how to preserve data freshness constitutes the primary challenge in designing caching policies for IoT sensing. A recently proposed performance criterion can be adopted to quantify data freshness, namely, *Age of Information* (AoI) [13]. The AoI of a content item is defined as the amount of time that has passed since the last measurement of this content item. Given the arrivals of user requests, cache update is needed to reduce the average AoI of caching content items [13]. Nevertheless, excessive cache updates will generate considerable energy consumption and challenge the battery life of sensors. Hence, these characteristics of IoT sensing require new and efficient caching policies in IoT sensing networks.

A. Related Work

AoI was initially investigated in [13] to evaluate status update for packet delivery between a source node and destination node. The author derived the average AoI by considering a simple queuing model. Such a source-destination scenario was further investigated in [14] under a more complex queuing system, i.e., $M/G/1$. The study in [15] also focused on this

Part of this work has been presented at the IEEE Globecom, 2020.

This work was supported in part by the U.S. National Science Foundation under Grant CCF-1908308, in part by Shenzhen Science and Technology Innovation Commission Grant R2020A045 and Canadian Natural Sciences and Engineering Research Council Grant RGPIN-2019-06348, in part by National NSFC Grant No. 61902044 and No. 61872184, and in part by Fundamental Research Funds for the Central Universities (Grant No. 2020CDJQY-A022). (*Corresponding author: Xiongwei Wu.*)

X. Wu and P. C. Ching are with the Department of Electronic Engineering, The Chinese University of Hong Kong, Shatin, Hong Kong SAR of China (e-mail: xwwu@ee.cuhk.edu.hk; pcching@ee.cuhk.edu.hk).

X. Li is with the School of Big Data & Software Engineering, Chongqing University, Chongqing, 401331 China, and also with the Key Laboratory of Dependable Service Computing in Cyber Physical Society (Chongqing University), Ministry of Education, China (email: lixiuhua1988@gmail.com)

J. Li is with the School of Electronic and Optical Engineering, Nanjing University of Science and Technology, Nanjing 210094, China. (e-mail: jun.li@njust.edu.cn).

V. C. M. Leung is with College of Computer Science & Software Engineering, Shenzhen University, Shenzhen 518060, China, and also with the Department of Electrical and Computer Engineering, The University of British Columbia, Vancouver, BC V6T 1Z4, Canada (email: vleng@ieee.org).

H. V. Poor is with the Department of Electrical Engineering, Princeton University, Princeton, NJ 08544. (e-mail: poor@princeton.edu).

scenario and investigated average and peak AoI. The authors in [16] studied the optimal policy for packet delivery from a source to a remote destination by considering an age penalty function. In general, these works were extensions of the study in [13], which characterized the average or peak AoI based on different types of queuing models. Later on, AoI was adopted to evaluate the performance of IoT sensing networks, which involved multiple sensors and error-prone wireless links. The study in [17] investigated the peak AoI under different scales of IoT networks by considering Bernoulli traffic. Taking into account sampling cost and update cost, the research in [18] examined update policies under a single sensor scenario and multiple sensors scenario, respectively. This line of research generally attempted to derive analytical expressions of age-based performance metrics under queuing models, and focused on performance analysis by utilizing optimization theory.

Some recent works have investigated intelligent policies for cost-effective caching in IoT sensing networks by applying reinforcement learning (RL). The authors in [19] considered a single sensor and proposed to minimize the average AoI subject to the average number of updates. Treating the model parameters of neural networks as transient content, the study in [20] proposed to minimize the average AoI plus cost by using deep Q-network (DQN). Both studies evaluated cache update cost by counting the number of content transmissions. The study in [21] proposed to minimize the average AoI by considering sensing and transmission energy cost. The authors in [22] considered cache update at multiple sensors, and investigated the tradeoff between energy consumption and AoI via Q-learning. Similarly, the tradeoff between AoI and energy consumption was also investigated in studies [12], [23]. Moreover, the studies in [24], [25] utilized actor-critic (AC)-based approaches to investigate how to cache transient content by considering content update cost. In our preliminary work [26], we proposed a DQN based cache updating design to decrease the average AoI as well as energy consumption given the distinct storage size of IoT sensing data, characteristics of wireless channels, and time-varying content popularity. All of these studies focused on cost-effective update policies at a single EN, which aggregates sensing data generated from sensors at its coverage only.

B. Contributions

This paper investigates intelligent policies of cost-effective cache update in the IoT sensing network, where multiple ENs cache sensing data under the coordination of the cloud. Compared with prior studies on a single EN that only entails sensors at its coverage [12], [19], [21]–[25], this work investigates a more general scenario. Specifically, each EN is likely to communicate with a subset of sensors because of the short communication ranges at IoT sensors [27]. Thus, each EN needs to upload content items generated from its coverage to the cloud so that other ENs beyond the communication range can download these content items and provide accessing services for data consumers. Consequently, cache update at multiple ENs not only requires energy consumption at sensors but also leads to fronthaul traffic loads. Given limited battery levels at sensors and capacity of fronthaul links in reality, it is imperative to find cache update policies that preserve data freshness while reducing update costs. For this reason, we consider a more integrative performance metric, involving the average AoI, energy consumption and fronthaul traffic loads. In addition, the average AoI in a multiple ENs scenario is characterized by user requests received at all ENs. We need to take into account space-time dynamics of content popularity [28] in comparison to the studies [12], [19], [21]–[23].

The considered scenario results in a multi-agent discrete decision-making, where the space of the discrete decisions grows exponentially versus system parameters, e.g., the number of ENs. Some conventional RL algorithms used in prior studies, e.g., DQN, suffer from high computational complexity and brittleness to scalability, which are not efficient in handling the multi-agent tasks [29]. We therefore devise a novel deep RL (DRL) approach with an output size linearly increases w.r.t. these system parameters. The proposed approach utilizes the idea of the state-of-the-art RL algorithm, similar to the soft actor-critic (SAC) in [30], that is originally applicable to continuous decision-making only.

The main contributions of this paper are summarized as follows.

- We investigate intelligent cache update design at multiple ENs in IoT sensing, which generalizes prior studies involving a single EN [12], [19], [21]–[25]. We first propose a multi-agent DRL framework to minimize the average AoI of caching content items plus cache update costs, i.e., transmission energy consumption and fronthaul traffic loads. We also derive a characterization for average transmission energy consumption at IoT sensors under an effective wireless transmission condition.
- To deal with the formulated problem, we devise a multi-agent discrete variant of SAC with an output size that linearly increases versus the numbers of ENs and IoT sensors. The core idea is that we customize the *Gumbel-SoftMax* (GS)-sampler to approximately generate differentiable actions. Meanwhile, the utilization of entropy regularization can effectively enhance *exploration*, which assists to prevent premature convergence.
- To reduce the communication overhead between ENs and the cloud, we further generalize the proposed centralized algorithm into decentralized control. Particularly, each EN serves as an independent agent with a decentralized policy, exploring its caching decisions based on local observations. We maintain the centralized soft Q-function at the cloud processor (CP) to augment EN coordination, favorably improving system reward.
- Simulation results are presented to demonstrate that the proposed RL approach outperforms existing RL-based caching policies, and unveil how transmission energy and fronthaul traffic load considerations compromise data freshness in IoT sensing networks.

The remainder of this paper is organized as follows. Section II introduces the system model. Section III describes the problem formulation. Section IV develops a centralized DRL-based cache update scheme, and Section V develops a

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCOMM.2021.3086535, IEEE Transactions on Communications

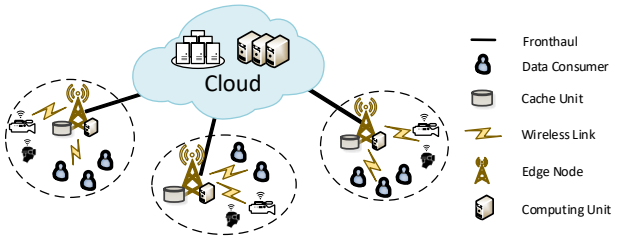


Fig. 1: Illustration of an IoT sensing network.

decentralized DRL-based cache update scheme. Section VI shows the performance evaluation, and Section VII concludes the paper.

II. System Model

As depicted in Fig. 1, consider an IoT sensing network, in which a total of B ENs are connected to the CP through wired fronthaul links. Every EN is equipped with a cache unit and a computing unit, which empower edge caching and edge computing, respectively. Consequently, such wireless networks allow ENs to serve as gateways between IoT sensors and data consumers [24]. More specifically, ENs are capable of caching sensing data generated by various kinds of sensors within their communication ranges. Meanwhile, data consumers can submit their requests to ENs and retrieve corresponding content for data processing and analysis. Notably, we do not assume any explicit roles for data consumers. They could be either static devices (e.g., computers) or mobile devices (e.g., smartphones, vehicles). For instance, data consumers can inspect temperatures or humidities of the environment on mobile applications. For ease of discussion, we assume that, each EN coordinates F sensors that are randomly distributed within its coverage. As such, there are a total of $B \times F$ sensors in the network. Each sensor is supposed to communicate with the nearest EN. Let $\mathcal{B} = \{1, 2, \cdots, B\}$ and $\mathcal{F} = \{1, 2, \cdots, B \times F\}$ denote the indices of ENs and sensors. Moreover, $\mathcal{F}_b = \{(b-1)F + 1, \cdots, bF\}$ denotes the indices of the sensors coordinated by the b-th EN. That is, $\mathcal{F} = \cup_{b \in \mathcal{B}} \mathcal{F}_b$.

A. Age of Information

The system operation time is assumed to be divided into epochs, i.e., $t = 1, 2, \cdots$. In IoT sensing, each content item cached at the EN, is generated by a certain IoT sensor. For instance, content item[1] $f \in \mathcal{F}$ implies that this content item is produced by the f-th sensor. In general, each caching content item can be temporally updated and replaced by a new version of the sensing data. We denote the generation epoch for the version of content item f cached at epoch t by v_f^t. Evidently, $v_f^t \leq t$. In addition, we assume that each content item should record its index and generation epoch, e.g., $\{f, v_f^t\}$, for data processing. To evaluate data freshness of a content item, we adopt AoI as the quality of service (QoS) metric, which counts how many epochs has passed since this content item was produced. In this way, the AoI of a caching content item f can be calculated as follows:

$$o_f^t = \max\{t - v_f^t, 1\}, \forall f \in \mathcal{F}, \tag{1}$$

which takes value from a finite range, i.e., $\{1, 2, \cdots T_{\max}\}$; and $T_{\max}$ denotes the upper limit, which implies the most outdated level of a content item [12], [23], [31]. We consider that each EN is able to receive user requests concerning content items generated by all of the sensors (e.g., $\forall f \in \mathcal{F}$). This is reasonable in real applications because data consumers usually have diverse preferences towards content items. Let $\{N_{f,b}^t\}_{f \in \mathcal{F}, b \in \mathcal{B}}$ be the number of user requests received by ENs at epoch t. Consequently, the average AoI to satisfy user demands at epoch t can be calculated as follows [20]:

$$O^t = \frac{\sum_{f \in \mathcal{F}, b \in \mathcal{B}} o_f^t N_{f,b}^t}{\sum_{f \in \mathcal{F}, b \in \mathcal{B}} N_{f,b}^t}. \tag{2}$$

As aforementioned, IoT sensing data are transient and gradually become stale as time passes.

B. Cache Update

To perform cache update, ENs should communicate with sensors through wireless links. Owing to channel fading, we assume the following successful transmission condition: data transmissions between IoT sensors and ENs are successful only on condition that the received SNR exceeds a predefined threshold η_{th}. Specifically, we assume that orthogonal channels are scheduled to different sensors. Thus, the received signal-to-noise (SNR) for sensor f delivering a content item to the associated EN can be expressed as:

$$\eta_f = \frac{P_f \chi_f^2 \kappa_f^2}{N_0 B_0}, \forall f \in \mathcal{F}, \tag{3}$$

where P_f is transmission power at sensor f; coefficient χ_f denotes the large-scale fading; N_0 denotes noise power spectrum density; and B_0 is the channel bandwidth. In addition, κ_f denotes the envelope of the small-scale fading, which is assumed to follow the Rayleigh distribution with probability density function $\mathbb{P}_{\kappa_f}(\kappa_f) = \frac{2\kappa_f}{\sigma} \exp(-\frac{\kappa_f^2}{\sigma})$, where $\sigma > 0$. Subsequently, the average transmission energy consumption for cache update, determined by channel gain and content size, is characterized as follows.

Proposition 1 *The average transmission energy $\bar{E}_f$ at the f-th IoT sensor ($\forall f \in \mathcal{F}$) for dispatching sensing data to the EN is as follows:*

$$\bar{E}_f = \frac{\ln 2 \times P_f s_f}{\ln 2 \times R_{th} \exp\left(-\frac{\eta_{th}}{2\beta_f}\right) + B_0 \exp\left(\frac{1}{2\beta_f}\right) \rho_f(\eta_{th} + 1)}, \tag{4}$$

where s_f is the storage size of content item f; function $\rho_f(\cdot)$ is defined as:

$$\rho_f(x) \triangleq \int_x^{\infty} \frac{1}{x} \exp(-x/(2\beta_f)) dx, \tag{5}$$

[1]By slightly abusing the notation, we denote the index of either a content item or an IoT sensor by f, and $f \in \mathcal{F}$.

and $\beta_f = \frac{P_f \chi_f^2 \sigma}{2N_0 B_0}$*; and* R_{th} *denotes the throughput threshold, i.e.:*

$$R_{th} \triangleq \log_2(1+\eta_{th}). \tag{6}$$

Proof. See Appendix A. ■

When a cached content item is updated, the associated EN should deliver the updated content item to the CP via fronthaul. Thus, when other ENs overhear requests that relate to sensors out of their coverage, they can fetch these content items from the CP[2]; however, this process causes many duplicate fronthaul transmissions, if these content items are repeatedly and asynchronously requested by users. To alleviate network congestions, we therefore allow each EN to pro-actively cache content items that are generated beyond their communication range and kept at the CP. Specifically, when a new version of content item $f \in \mathcal{F}_b$ is uploaded to the CP, other ENs (i.e., $\forall b' \neq b$) should further prefetch this content item with the cost of additional fronthaul traffic loads $(B-1)s_f$. We consider that the storages of cache units in ENs are sufficiently large enough to aggregate content items generated by IoT sensors (e.g., $\forall f \in \mathcal{F}$) in the network because the storage size of sensing data is often at a small size in practice. In addition, by using high-speed optical fiber, we assume that data transfer between the CP and ENs can be completed in a short time [32], which is negligible compared with the duration of each epoch. The main notation used in the paper is summarized in Table I.

TABLE I: Notation

Notation	Description
$\boldsymbol{a}_b$	Local action at EN b
B	Number of ENs
F	Number of sensors
$\bar{E}_f$	Average transmission energy for delivering content item f
$\mathcal{F}_b$	Indices of sensors coordinated by EN b
$N_{f,b}$	Number of requests for content item f received at EN b
o_f^t	AoI of content item f at epoch t
$\bar{O}^t$	Average AoI to satisfy user demands at epoch t
P_f	Transmission power at the f-th sensor
$Q(\boldsymbol{s},\boldsymbol{a})$	Q-function
r	Reward
$\boldsymbol{s}_b$	Local observation of EN b
ω_1, ω_2	Importance factors
$\pi(\cdot\|\boldsymbol{s})$	Policy function
γ	Discount factor
$\mathcal{H}(\cdot)$	Entropy operator

As previously stated, since battery levels of sensors and capacity of fronthaul links are restricted in reality, caching content should be reasonably updated to achieve a favorable tradeoff among average AoI, energy consumption, and fronthaul traffic load. In view of this, we formulate a cache update decision-making strategy in the next section.

III. MDP Problem Formulation

Our goal is to find cache update policies, which allow ENs to reasonably update content items under different states, minimizing the long-term average weighted cost. This weighed cost is supposed to comprise average AoI, transmission energy, and fronthaul traffic loads.

A. Multi-Agent Cooperative MDP

The average AoI (e.g., (2)) depends critically on the content popularity distribution at ENs:

$$\hat{p}_{f,b}^t = N_{f,b}^t / \textstyle\sum_{f' \in \mathcal{F}, b \in \mathcal{B}} N_{f',b}^t, \forall f, b, \tag{7}$$

which is usually temporally evolving and can be modeled as a Markov chain [28]. Moreover, updating a caching content item changes the corresponding AoI state, which influences the decision-making at the subsequent epochs. In such a dynamic environment, cache update decisions are dependent across consecutive epochs. In view of this, we model cache updating at multiple ENs as a multi-agent cooperative MDP, where all agents collaborate to optimize a common long-term objective (e.g., the long-term weighted cost) based on local observations. Particularly, every EN serves as an agent, and we define the basic elements of a multi-agent MDP as follows.

- **State:** we denote state space by $\mathcal{S}$, which contains all possible states $\boldsymbol{s}$. Every state $\boldsymbol{s}$ consists of local observations of all agents, i.e., $\boldsymbol{s} = \{\boldsymbol{s}_1, \boldsymbol{s}_2, \cdots, \boldsymbol{s}_B\}$. In the IoT sensing network, each agent (i.e., EN) is capable of observing the AoI of every content item and local user requests. Then, the local state of EN b is defined as follows:

$$\boldsymbol{s}_b^t = \left(\{o_f^t\}_{f\in\mathcal{F}}, \{\hat{p}_{f,b}^t\}_{f\in\mathcal{F}}\right), \forall b \in \mathcal{B}, \tag{8}$$

  which is a $2BF$-dimensional tuple.
- **Action:** Let $\boldsymbol{a} = \{\boldsymbol{a}_1, \boldsymbol{a}_2, \cdots, \boldsymbol{a}_B\}$ denote a joint action, where local action $\boldsymbol{a}_b$ implies which content item[3] should be selected to update at EN b. Accordingly, the local action space of agent b can be given by $\mathcal{A}_b = \{0\} \cup \mathcal{F}_b$; and the joint action space is given by:

$$\mathcal{A} = \cup_{b\in\mathcal{B}} \mathcal{A}_b. \tag{9}$$

  Particularly, when local action $\boldsymbol{a}_b = 0$, it implies that EN b remains idle and presents null transmission energy and traffic loads. Otherwise, the corresponding sensor needs to upload the current measurement of content item $\boldsymbol{a}^t \in \mathcal{F}$ into EN b. That is,

$$\begin{aligned} o_f^{t+1} = \min\big\{&(o_f^t+1)\times(1-\mathcal{I}(f,\boldsymbol{a}_b^t)) \\ &+\mathcal{I}(f,\boldsymbol{a}_b^t), T_{\max}\big\}, \forall f \in \mathcal{F}_b, \forall b \in \mathcal{B}, \end{aligned} \tag{10}$$

  where $\mathcal{I}(\cdot)$ is an indicator function[4]. After each agent takes action $\boldsymbol{a}_b^t$, the system state becomes $\boldsymbol{s}^{t+1}$ with transition probability $\Pr\{\boldsymbol{s}^{t+1}|\boldsymbol{s}^t, \boldsymbol{a}^t\}$ at epoch $t+1$.
- **Reward:** In a multi-agent cooperative MDP, all agents are expected to share a common reward r^{t+1}, which unveils how effective a joint action $\boldsymbol{a}^t$ is [29]. Recall that our objective is to minimize the average AoI whilst

[2]In some IoT applications such as collaborative mobile sensing, message passing among ENs can be completed by fulfilling the maximum mutual information or minimum entropy to further reduce fronthaul traffic load.

[3]The proposed framework can also be generalized to case where multiple content items are determined to update at each EN. This is, however, at the cost of larger system bandwidth and energy consumption.

[4]Given parameters x, y, when $x = y$, we have $\mathcal{I}(x,y) = 1$; otherwise, $\mathcal{I}(x,y) = 0$.

reducing transmission energy consumption at sensors and network congestions. Consequently, we define the average weighted cost at each epoch as follows:

$$C^{t+1} = \frac{\sum_{f\in\mathcal{F},b\in\mathcal{B}} o_f^{t+1} N_{f,b}^{t+1}}{\sum_{f\in\mathcal{F},b\in\mathcal{B}} N_{f,b}^{t+1}} + \omega_1 \sum_{b\in\mathcal{B}} \bar{E}_f\big|_{f=\boldsymbol{a}_b^t} + \omega_2 \sum_{b\in\mathcal{B}} (B-1) s_f\big|_{f=\boldsymbol{a}_b^t}, \quad (11)$$

where the first term on the right-hand side is the average AoI to satisfy user demands arrived at epoch $t+1$; $\bar{E}_f\big|_{f=\boldsymbol{a}_b^t}$ stands for the average energy consumption of uploading the selected content item $\boldsymbol{a}_b^t$ to its associated EN, and $(B-1)s_f\big|_{f=\boldsymbol{a}_b^t}$ denotes the additional fronthaul traffic loads for delivering content item f to other ENs from the CP; ω_1 and ω_2 are non-negative coefficients to weigh the importance of energy and traffic cost. For notational convenience, we define $\bar{E}_0 = 0$ and $s_0 = 0$. To reduce the average weighted cost, we design the reward as $r^{t+1} \triangleq R(\boldsymbol{s}^{t+1}, \boldsymbol{s}^t, \boldsymbol{a}^t)$, where

$$R(\boldsymbol{s}^{t+1}, \boldsymbol{s}^t, \boldsymbol{a}^t) = -C^{t+1}, \quad (12)$$

which is a negative value.

Consequently, we aim to find a caching policy π^*, which is able to generate a joint action $\boldsymbol{a}$ given any state $\boldsymbol{s}$ that maximizes the expected discounted cumulative reward as follows:

$$\pi^* = \arg\max_{\pi} \mathbb{E}[V^t|\pi], \quad (13)$$

where

$$V^t = \sum_{\tau=0}^{\infty} (\gamma)^\tau r^{t+\tau+1}, \quad (14)$$

and $\gamma \in [0,1)$ is a discounted factor.

To address problem (13), one can resort to model-based approaches [33], which usually rely on the knowledge of $\Pr\{\boldsymbol{s}^{t+1}|\boldsymbol{s}^t, \boldsymbol{a}^t\}$. However, transition probability is usually uncertain and difficult to estimate. Even if we could have this knowledge, our problem is still intractable due to the curse of dimensionality in multi-agent settings. These challenges motivate us to explore data-driven approaches, i.e., RL, which is a consequence of properly utilizing past experiences. In the following sections, we develop efficient RL algorithms to handle the formulated problem.

IV. Centralized Multi-Agent Discrete Soft Actor-Critic-Based Caching

In this section, we develop a centralized RL algorithm for cache update in IoT sensing network, where the CP acts as the centralized agent and coordinates caching decisions for all ENs. We first outline the background of RL and identify the challenges of conventional approach. Then, we devise an efficient RL approach, which is a multi-agent discrete variant of the state-of-the-art RL.

A. Background of Reinforcement Learning

Canonical RL generally aims to estimate the following Q-function:

$$Q^*(\boldsymbol{s}, \boldsymbol{a}) = [V^t|\boldsymbol{s}^t = \boldsymbol{s}, \boldsymbol{a}^t = \boldsymbol{a}, \pi^*], \quad (15)$$

which indicates the expected cumulative reward after taking action $\boldsymbol{a}^t$ under state $\boldsymbol{s}^t$, subsequently following policy π^*; and V^t is defined in (14). An optimal policy can be characterized by the following *Bellman Optimality*.

Lemma 1 *An optimal policy π^* leads to the following recursive equations [33]:*

$$Q^*(\boldsymbol{s}, \boldsymbol{a}) = \mathbb{E}_{\boldsymbol{s}'}\big[R(\boldsymbol{s}', \boldsymbol{s}, \boldsymbol{a}) + \gamma \max_{\boldsymbol{a}'\in\mathcal{A}} Q^*(\boldsymbol{s}', \boldsymbol{a}')|\boldsymbol{s}, \boldsymbol{a}\big], \quad (16)$$

where $R(\boldsymbol{s}', \boldsymbol{s}, \boldsymbol{a})$ is the reward function; $\boldsymbol{s}'$ is the subsequent state after taking action $\boldsymbol{a}$ under state $\boldsymbol{s}$; $Q^*(\boldsymbol{s}, \boldsymbol{a})$ denotes the Q-function by following π^*.

Lemma 1 lays the foundation for DQN. As a popular approach for discrete decision-marking, DQN utilizes deep neural networks (DNNs) as function approximators to predict the optimal Q-function. Readers are referred to [34] for more detail. By adopting this approach, an optimal policy can be given by the following mapping function:

$$\pi^*(\boldsymbol{s}) \triangleq \max_{\boldsymbol{a}\in\mathcal{A}} Q^*(\boldsymbol{s}, \boldsymbol{a}). \quad (17)$$

In other words, DQN needs to output the values of Q-function over all possible discrete actions (i.e., $\boldsymbol{s} \to \mathbb{R}^{|\mathcal{A}|}$). This practice results in slow convergence and brittleness to scalability. Therefore, DQN is difficult to be applied in multi-agent and high dimensional settings [30].

B. Proposed Multi-Agent Discrete Soft Actor-Critic Learning

SAC is the state-of-the-art RL algorithm, which is a result of an entropy regularized formalism that augments *exploration* [30]. This approach entails an AC framework, which specifies the stochastic policy and soft Q-function separately. It attempts to find a stochastic policy that maximizes the expected cumulative reward while taking as many diverse actions as possible. Consequently, SAC can achieve high sample efficiency [30]. However, the SAC in [30] is only applicable in continuous settings. We now develop a multi-agent discrete variant of SAC learning, which is suitable to handle discrete decision-making especially in high dimensional settings.

Similarly, our objective is to find a stochastic policy $\pi(\boldsymbol{a}|\boldsymbol{s})$ that maximizes the expected cumulative reward plus its entropy, i.e.:

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{\{\boldsymbol{s}^t, \boldsymbol{a}^t\}} \left[\sum_{t=0}^{+\infty} (\gamma)^t \Big(r^{t+1} + \alpha \mathcal{H}(\pi(\cdot|\boldsymbol{s}^t))\Big)\right], \quad (18)$$

where $\pi(\cdot|\boldsymbol{s}^t)$ is a categorical distribution indicating the probability of taking any action under state $\boldsymbol{s}^t$; the entropy term is defined as $\mathcal{H}(\pi(\cdot|\boldsymbol{s}^t)) \triangleq \mathbb{E}_{\boldsymbol{a}}\big[-\log(\pi(\boldsymbol{a}|\boldsymbol{s}^t))\big]$; α is the temperature parameter and controls the magnitude of entropy

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCOMM.2021.3086535, IEEE Transactions on Communications

regularization. In general, a larger α prompts the agents to carry out a more random *exploration* in decision making.

Accordingly, the soft Q-function can be defined as follows:

$$Q(\boldsymbol{s},\boldsymbol{a}) = \mathbb{E}\left[V^t + \alpha\sum_{\tau=1}^{+\infty}(\gamma)^\tau \mathcal{H}(\pi(\cdot|\boldsymbol{s}^{t+\tau}))\big|\boldsymbol{s}^t=\boldsymbol{s},\boldsymbol{a}^t=\boldsymbol{a}\right], \tag{19}$$

where V^t is the discounted cumulative reward, given by (14). This further gives rise to the soft value function

$$V(\boldsymbol{s}) = \mathbb{E}_{\boldsymbol{a}\sim\pi(\cdot|\boldsymbol{s})}\left[Q(\boldsymbol{s},\boldsymbol{a}) - \alpha\log\pi(\boldsymbol{a}|\boldsymbol{s})\right], \tag{20}$$

where the expectation is taken over an action $\boldsymbol{a}$ which is drawn from the distribution $\pi(\cdot|\boldsymbol{s})$.

According to Lemma 1, we have the similar recursive equality as follows:

$$Q(\boldsymbol{s}^t,\boldsymbol{a}^t) = \mathbb{E}_{\boldsymbol{s}^{t+1}}\left[r^{t+1} + \gamma V(\boldsymbol{s}^{t+1})\right]. \tag{21}$$

To pave the way for the multi-agent discrete SAC (MADSAC), we introduce the *Soft Policy Improvement* [30].

Lemma 2 *Given a policy π_{old} and soft Q-function $Q_{old}(\boldsymbol{s},\boldsymbol{a})$ with a finite action space $\mathcal{A}$, a new policy π_{new} is given by:*

$$\min_{\pi'} D_{KL}\Big(\pi'(\cdot|\boldsymbol{s})\|\exp(Q_{old}(\boldsymbol{s},\cdot)/\alpha)/Z(\boldsymbol{s})\Big), \forall \boldsymbol{s}\in\mathcal{S}, \tag{22}$$

where $D_{KL}(\cdot\|\cdot)$ denotes the Kullback-Leibler divergence[5]*, and the normalization factor $Z(\boldsymbol{s}) = \sum_{\boldsymbol{a}}\exp(Q_{old}(\boldsymbol{s},\boldsymbol{a})/\alpha)$. Then, it leads to $Q_{new}(\boldsymbol{s},\boldsymbol{a}) \geq Q_{old}(\boldsymbol{s},\boldsymbol{a})$ for any $(\boldsymbol{s},\boldsymbol{a}) \in \mathcal{S}\times\mathcal{A}$ [30], [35].*

Following the elementary steps of the SAC in [30], we consider a parameterized $Q_{\boldsymbol{\phi}}(\boldsymbol{s},\boldsymbol{a})$ and policy $\pi_{\boldsymbol{\theta}}(\cdot|\boldsymbol{s})$, where $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$ are parameters of some function approximators, e,g., DNNs. Since the soft value function can be expressed by (20), we do not incorporate a separate function approximator here. In addition, $Q_{\boldsymbol{\phi}}(\boldsymbol{s},\boldsymbol{a})$ denotes a mapping with unit output, i.e., $(\boldsymbol{s},\boldsymbol{a}) \to \mathbb{R}$, instead of the number of all possible discrete actions. This is because we can solely use policy function to generate decisions.

To confine the output of stochastic policy $\pi_{\boldsymbol{\theta}}(\cdot|\boldsymbol{s})$ in multi-agent RL, we first consider the following decomposition:

$$\pi_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s}) = \Pi_{b\in\mathcal{B}}\ \mu_{\boldsymbol{\theta}}(\boldsymbol{a}_b|\boldsymbol{s}), \forall \boldsymbol{a}\in\mathcal{A}, \boldsymbol{s}\in\mathcal{S}, \tag{23}$$

where $\boldsymbol{a} = \{\boldsymbol{a}_1,\boldsymbol{a}_2,\cdots,\boldsymbol{a}_B\}$. Accordingly, we can devise a single function approximator to simultaneously learn the probabilities of taking local actions, e.g., $\boldsymbol{a}_b \in \mathcal{A}_b, \forall b\in\mathcal{B}$, with output dimension $\sum_{b\in\mathcal{B}}|\mathcal{A}_b|$. In addition, $\sum_{\boldsymbol{a}_b\in\mathcal{A}_b}\mu_{\boldsymbol{\theta}}(\boldsymbol{a}_b|\boldsymbol{s}) = 1$, for $\forall b\in\mathcal{B}$. In this way, the decisions of the individual agents can be jointly made by a (centralized) agent conditioned on given state $\boldsymbol{s}$.

Then, on the basis of (20)-(22), we train the parameters of the above functions, $\{\boldsymbol{\theta},\boldsymbol{\phi}\}$, by using historical experiences, e.g., $\xi^t = (\boldsymbol{s}^t,\boldsymbol{a}^t,r^{t+1},\boldsymbol{s}^{t+1})$. We use the *Replay Buffer (RB)* to store some recent experiences, i.e., $\xi^t \in \Xi$. Specifically, we can train the policy function according to *Soft Policy Improvement*. By omitting the normalization factor $Z(\cdot)$ in (22), the policy parameter (i.e., $\boldsymbol{\theta}$) can be trained by adopting stochastic gradient descent to minimize the following loss function:

$$J_\pi(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{s}^t\sim\Xi}\left[\mathbb{E}_{\boldsymbol{a}\sim\pi_{\boldsymbol{\theta}}}[\alpha\log(\pi_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s}^t)) - Q_{\boldsymbol{\phi}}(\boldsymbol{s}^t,\boldsymbol{a})]\right], \tag{24}$$

where the expectation over $\boldsymbol{s}^t$ can be approximated by drawing samples from the *RB*. However, (24) incorporates an expectation over actions following policy distribution $\pi_{\boldsymbol{\theta}}(\cdot|\boldsymbol{s}^t)$. A challenging issue is that the gradient w.r.t. $\boldsymbol{\theta}$ can not be backpropagated in a normal manner if we directly utilize $\pi_{\boldsymbol{\theta}}(\cdot|\boldsymbol{s}^t)$ to generate samples. To deal with it, we resort to the *reparameterization trick* [36].

The core idea of our approach is to find action samples that are differentiable w.r.t. the policy parameter $\boldsymbol{\theta}$. To proceed, we first introduce auxiliary random variables $\{g_{i,b}, \forall i\in\mathcal{A}_b, b\in\mathcal{B}\}$, which are i.i.d. and follow the Gumbel distribution, e.g., $Gumbel(x) = \exp(-(x+\exp(-x)))$. With these Gumbel-distributed variables, we arrive at the following lemma.

Lemma 3 *Given a stochastic policy $\pi_{\boldsymbol{\theta}}$ for a multi-agent discrete decision-making, a joint action $\hat{\boldsymbol{a}} = \{\hat{\boldsymbol{a}}_1,\hat{\boldsymbol{a}}_2,\cdots,\hat{\boldsymbol{a}}_B\}$ can be generated as follows:*

$$\hat{\boldsymbol{a}}_b = \arg\max_{i\in\mathcal{A}_b}\ [g_{i,b} + \log\mu_{\boldsymbol{\theta}}(i|\boldsymbol{s})], \forall b\in\mathcal{B}. \tag{25}$$

That is, $Pr\{\hat{\boldsymbol{a}}|\boldsymbol{s}\} = \pi_{\boldsymbol{\theta}}(\hat{\boldsymbol{a}}|\boldsymbol{s})$.

Proof. See Appendix B. ■

The above lemma is an extension of the results in [36]. The Gumbel distribution is commonly used to model the maximum of a group of samples drawn from some distribution. By using Gumbel-distributed variables $\{g_{i,b}\}$, we can always calculate an action sample $\hat{\boldsymbol{a}}$ via (25) with the probability $\pi_{\boldsymbol{\theta}}(\hat{\boldsymbol{a}}|\boldsymbol{s})$. Consequently, the sample action $\hat{\boldsymbol{a}}$ is a function of the policy parameter $\boldsymbol{\theta}$. Owing to the involvement of $\arg\max$, $\hat{\boldsymbol{a}}$ attains an ordinary value that is non-differentiable w.r.t. $\boldsymbol{\theta}$. For this reason, we still cannot perform stochastic gradient descent. To cope with this issue, the $\arg\max$ in (25) can be further approximated by a smooth operation, e.g., *SoftMax*, resulting in a GS-sampler.

Corollary 1 *Suppose vector $\boldsymbol{z}_b = [z_{i,b}] \in \mathbb{R}^{|\mathcal{A}_b|}, \forall b\in\mathcal{B}$, where each element is given by*

$$z_{i,b} = \frac{\exp((g_{i,b}+\log(\mu_{\boldsymbol{\theta}}(i|\boldsymbol{s})))/c_0)}{\sum_{j\in\mathcal{A}_b}\exp((g_{j,b}+\log(\mu_{\boldsymbol{\theta}}(j|\boldsymbol{s})))/c_0)}, \forall i\in\mathcal{A}_b, \forall b\in\mathcal{B}, \tag{26}$$

where the SoftMax coefficient $c_0 > 0$. Then, when c_0 goes to 0, $\boldsymbol{z}_b$ approaches a unit vector with one element being 1 and all other elements being 0, $\forall b\in\mathcal{B}$.

The motivation of Corollary 1 is to encode ordinary and non-differentiable actions calculated via (25) to unit vectors approximately, which are differentiable w.r.t. $\boldsymbol{\theta}$. As such, we are capable to generate action samples from a categorical distribution $\boldsymbol{\pi}_{\boldsymbol{\theta}}(\cdot|\boldsymbol{s})$ via GS-sampler, i.e., $GS(\boldsymbol{\pi}_{\boldsymbol{\theta}}(\cdot|\boldsymbol{s}))$. Toward

[5]Given two categorical distributions π_1,π_2, we have $D_{KL}(\pi_1\|\pi_2) = \sum_x \pi_1(x)\log\left(\frac{\pi_1(x)}{\pi_2(x)}\right)$.

Algorithm 1 Centralized Multi-Agent Discrete SAC-Based Cache Update

1: Initialize soft Q-function parameters $\boldsymbol{\phi}_1, \boldsymbol{\phi}_2$, policy parameter $\boldsymbol{\theta}$
2: Initialize parameters of target networks $\boldsymbol{\phi}_i^- \leftarrow \boldsymbol{\phi}_i, i = 1, 2$
3: **for** $t = 0, 1, 2, \cdots$ **do**
4: Observe $\boldsymbol{s}$ and take action $\boldsymbol{a} \sim \pi_{\boldsymbol{\theta}}(\cdot|\boldsymbol{s})$
5: Observe $\boldsymbol{s}'$ and r
6: Store $\xi = (\boldsymbol{s}, \boldsymbol{a}, r, \boldsymbol{s}')$ in *RB*
7: **procedure** TRAINMADSAC
8: Randomly draw a batch of N experiences as Ξ_N
9: **for** each $\xi = (\boldsymbol{s}, \boldsymbol{a}, r, \boldsymbol{s}') \in \Xi_N$ **do**
10: Calculate target values: $y_\xi = r + \gamma(\min_{i=1,2} Q_{\boldsymbol{\phi}_i^-}(\boldsymbol{s}', \boldsymbol{a}') - \alpha \log \pi_{\boldsymbol{\theta}}(\boldsymbol{a}'|\boldsymbol{s}'))$, where $\boldsymbol{a}' = GS(\pi_{\boldsymbol{\theta}}(\cdot|\boldsymbol{s}'))$
11: **end for**
12: Update $\boldsymbol{\phi}_i$ by taking a one-step gradient descent of $\frac{1}{N}\sum_{\xi\in\Xi_N}(Q_{\boldsymbol{\phi}_i}(\boldsymbol{s}, \boldsymbol{a}) - y_\xi)^2$, for $i = 1, 2$
13: Update $\boldsymbol{\theta}$ by taking a one-step gradient descent of $\frac{1}{N}\sum_{\xi\in\Xi_N}\big(\min_{i=1,2} Q_{\boldsymbol{\phi}_i}(\boldsymbol{s}, GS(\pi_{\boldsymbol{\theta}}(\cdot|\boldsymbol{s}))) - \alpha \log \pi_{\boldsymbol{\theta}}(GS(\pi_{\boldsymbol{\theta}}(\cdot|\boldsymbol{s}))|\boldsymbol{s})\big)$
14: Update temperature α by taking a one-step gradient descent of $-\frac{1}{N}\sum_{\xi\in\Xi_N}\alpha\big(\log \pi_{\boldsymbol{\theta}}(GS(\pi_{\boldsymbol{\theta}}(\cdot|\boldsymbol{s}))|\boldsymbol{s}) - \bar{H}\big)$
15: Update parameters of target networks $\boldsymbol{\phi}_i^- \leftarrow \tau\boldsymbol{\phi}_i + (1-\tau)\boldsymbol{\phi}_i^-$, for $i = 1, 2$
16: **end procedure**
17: **end for**

this end, the policy parameter $\boldsymbol{\theta}$ can be updated by utilizing stochastic gradient descent to minimize loss function $J_\pi(\boldsymbol{\theta})$.

Regarding the updates of soft Q-function and temperature parameter α, the procedure exactly follows the steps in [30]. On the basis of Bellman optimality and (21), the parameter of the soft Q-function can be updated by minimizing the following loss term [30]:

$$J_Q(\boldsymbol{\phi}) = \mathbb{E}_{\xi^t\sim\Xi}\left[\left(Q_{\boldsymbol{\phi}}(\boldsymbol{s}^t, \boldsymbol{a}^t) - (r^{t+1} + \gamma\bar{V}(\boldsymbol{s}^{t+1}))\right)^2\right], \tag{27}$$

where the target value $\bar{V}(\boldsymbol{s}^{t+1})$ is given by:

$$\mathbb{E}_{\boldsymbol{a}\sim\pi_{\boldsymbol{\theta}}}\left[Q_{\boldsymbol{\phi}^-}(\boldsymbol{s}^{t+1}, \boldsymbol{a}) - \alpha \log \pi_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s}^{t+1})\right], \tag{28}$$

and $Q_{\boldsymbol{\phi}^-}(\boldsymbol{s}, \boldsymbol{a})$ is the target network with parameter $\boldsymbol{\phi}^-$ being updated in the *soft copy* manner. That is, $\boldsymbol{\phi}^- \leftarrow \tau\boldsymbol{\phi} + (1-\tau)\boldsymbol{\phi}^-$, where step size τ is a small positive value to stabilize network training [37]. The temperature parameter α is updated by minimizing the following term:

$$J(\alpha) = \mathbb{E}_{\boldsymbol{s}^t\sim\Xi}\left[\mathbb{E}_{\boldsymbol{a}\sim\pi}[-\alpha(\log(\pi_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s}^t)) - \bar{H})]\right], \tag{29}$$

where the target entropy $\bar{H}$ is a constant and denotes the desired expected entropy [30].

C. Algorithm Implementation

We present the centralized algorithm implementation for cache update in IoT sensing network, i.e., MADSAC-centralized control (MADSAC-CC). A workflow is illustrated in Fig. 2a. The CP plays a role of the centralized agent and trains a centralized policy for all ENs. Accordingly, each EN should first transfer its local observations (e.g., $\boldsymbol{s}_b^t, \forall b \in \mathcal{B}$) to the CP every epoch. After collecting local observations, the CP returns local actions to each EN separately (e.g., $\boldsymbol{a}_b^t, \forall b \in \mathcal{B}$). Hereunder, we describe network design and algorithm implementation.

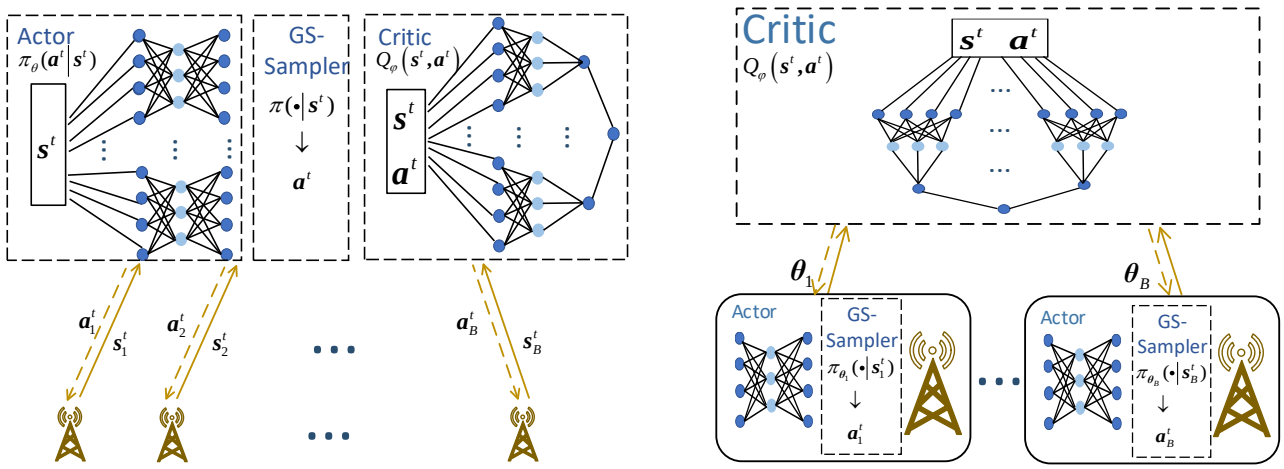
Network Design: Notably, the update of the soft Q-function entails bootstrapping (e.g., (27) and (28)), which inevitably leads to an overestimation issue [38]. To overcome this challenge, we adopt a clipped double Q-learning approach, where two separate Q-networks are concurrently trained, and the minimum of the outputs produced by the two networks is the estimate of the soft Q-function. This approach can reduce the variance of the overestimation error and offer stable target values to update policy parameters [38]. Subsequently, we denote the parameterized Q-functions by $Q_{\boldsymbol{\phi}_1}(\boldsymbol{s}, \boldsymbol{a})$ and $Q_{\boldsymbol{\phi}_2}(\boldsymbol{s}, \boldsymbol{a})$. Note that, the joint action $\boldsymbol{a}$ consists of a number of B unit vectors. We further maintain a policy network to output a $B(F+1)$-dimensional vector, where each element respectively corresponds to $\Pr\{\boldsymbol{a}_b|\boldsymbol{s}^t\}, \forall \boldsymbol{a}_b \in \mathcal{A}_b, \forall b \in \mathcal{B}$.

Algorithm Training: The CP maintains the *RB*, which has a finite capacity and can store the recent experiences for network training. Once the *RB* is fully loaded, we should replace the most outdated experience ξ^t by the latest one. The concrete implementation steps are shown in Algorithm 1. At every iteration, we need to randomly draw a mini-batch of N samples to approximate the expectation terms in $J_\pi(\boldsymbol{\theta}), \{J_Q(\boldsymbol{\phi}_i)\}$ and $J(\alpha)$. Particularly, steps 12 - 14 account for the updates of the soft Q-function parameters $\boldsymbol{\phi}_i, i = 1, 2$. We estimate the target values $\mathbb{E}_{\boldsymbol{a}\sim\pi_{\boldsymbol{\theta}}}[Q_{\boldsymbol{\phi}_i^-}(\boldsymbol{s}^{t+1}, \boldsymbol{a}) - \alpha \log \pi_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s}^{t+1})]$ in $J_Q(\boldsymbol{\phi}_i)$ by steps 11-13. Steps 15 - 16 carry out the updates of $\boldsymbol{\theta}$ and α, where action samples in $J_\pi(\boldsymbol{\theta})$ and $J(\alpha)$ are produced by the GS-sampler. All parameters, i.e., $\{\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \boldsymbol{\theta}, \alpha\}$, are trained by using stochastic gradient descent with proper learning rates.

D. Computational Complexity Analysis

The computational complexity is primarily determined by the network architectures of the Q-networks and policy networks. Suppose a total of H_c fully connected neural networks are being used as the hidden layers for the centralized soft Q-function, where the h-th hidden layer contains n_h^c neurons. The number of neurons in the input layer is specified by the dimension of the state and joint action, which is $B(BF+2F+1)$. The number of neurons in the output layer is 1. Therefore, the numbers of weights in the input layer, the h-th $(2 \le h \le H_c - 1)$ hidden layer, and the final hidden layer in turn are $B(BF+2F+1)n_1^c$, $n_{h-1}^c n_h^c$ and $n_{H_c}^c$. Regarding the policy network, we adopt H_a hidden fully connected layers, where the h-th hidden layer contains n_h^a neurons. Thus, the numbers of weights in the input layer, the h-th $(2 \le h \le H_a - 1)$ hidden layer and the final hidden layer are $BF(B+1)n_1^a$, $n_{h-1}^a n_h^a$ and $B(F+1)n_{H_a}^a$, respectively. Suppose that the computational complexity to train a single weight is W. Accordingly, the computational complexity of MADSAC-CC is $\mathcal{O}(W[B(BF+2F+1)n_1^c + \sum_{h=2}^{H_c} n_{h-1}^c n_h^c + n_{H_c}^c + BF(B+1)n_1^a + B(F+1)n_{H_a}^a + \sum_{h=2}^{H_a} n_{h-1}^a n_h^a])$,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCOMM.2021.3086535, IEEE Transactions on Communications



(a) Proposed centralized control.

(b) Proposed decentralized control.

Fig. 2: Diagrams of algorithm implementations.

which increases linearly with the number of sensors, F, and quadratically with the number of ENs, B.

In comparison, the conventional DQN entails a Q-network that contains $(F+1)^B$ neurons in the output layer. We consider H fully connected neural networks being implemented, where the h-th layer contains n_h neurons. Similar to the above analysis, the computational complexity is given by $\mathcal{O}(W[BF(B+1)n_1+\sum_{h=2}^{H} n_{h-1}n_h+(F+1)^B n_H])$, which grows polynomially with the number of sensors and exponentially with the number of ENs.

Remark 1 *In a nutshell, the key features of the proposed approach are the utilization of a GS-sampler and entropy regularization. On the one hand, by approximately encoding ordinary values of local actions into unit vectors, the output of the policy network is $B(F+1)$, which significantly reduces the size of the neural networks. On the other hand, the entropy-regularized formalism used in the proposed approach can enhance exploration and improve performance. That is, one can efficiently gather enough experiences to infer better actions compared with conventional RL algorithms [30], [33].*

It is worth mentioning that, when policy network is well-tuned, it can be solely utilized to select actions for all ENs. However, the centralized control essentially requires the CP to first aggregate local observations at ENs and then distribute caching decisions back to ENs at every decision-making epoch, which inevitably introduces very high communication overhead.

V. Decentralized Multi-Agent Discrete Soft Actor-Critic-Based Caching

To reduce communication overhead suffered from the centralized control, we develop a decentralized MADSAC-based caching scheme. Particularly, we devise decentralized policies for each agent to locally generate caching decisions, while utilizing the centralized soft-Q function to globally optimize these decentralized policies.

A. Proposed Decentralized Multi-Agent Discrete Soft Actor-Critic Learning

To generalize the proposed discrete variant of the SAC into the decentralized control, we maintain B parameterized stochastic policies for each agent, i.e., $\pi_{\boldsymbol{\theta}_b}(\cdot|\boldsymbol{s}_b)$, where $\boldsymbol{\theta}_b$ denotes the parameter of the corresponding function approximator, $\forall b \in \mathcal{B}$. According to the principle of the SAC, the decentralized MADSAC learning attempts to optimize the following entropy-regularized problem:

$$\max_{\{\boldsymbol{\theta}_b\}} \mathbb{E}_{\{\boldsymbol{s}_b^t\},\{\boldsymbol{a}_b^t\}}\left[\sum_{t=0}^{+\infty}(\gamma)^t\left(r^{t+1}+\alpha\sum_{b\in\mathcal{B}}\mathcal{H}(\pi_{\boldsymbol{\theta}_b}(\cdot|\boldsymbol{s}_b^t))\right)\right]. \tag{30}$$

Thus, the centralized soft-Q function can be defined as:

$$Q(\{\boldsymbol{s}_b\},\{\boldsymbol{a}_b\}) = \mathbb{E}\left[V^t+\sum_{\tau=1}^{+\infty}\sum_{b\in\mathcal{B}}(\gamma)^\tau\alpha\mathcal{H}(\pi_{\boldsymbol{\theta}_b}(\cdot|\boldsymbol{s}_b^{t+\tau}))\big|\boldsymbol{s}_b^t=\boldsymbol{s}_b,\boldsymbol{a}_b^t=\boldsymbol{a}_b,\forall b\in\mathcal{B}\right]. \tag{31}$$

Accordingly, we further maintain a parameterized soft Q-function $Q_{\boldsymbol{\phi}}(\{\boldsymbol{s}_b\},\{\boldsymbol{a}_b\})$, which can be applied to refine local (decentralized) policies. As such, policy parameters $\{\boldsymbol{\theta}_b\}$ can be trained by minimizing the following loss function:

$$J_\pi(\{\boldsymbol{\theta}_b\}) = \mathbb{E}_{\{\boldsymbol{s}_b^t\}\sim\Xi}\Bigg[\mathbb{E}_{\boldsymbol{a}_b\sim\pi_{\boldsymbol{\theta}_b}}\Bigg[\sum_{b\in\mathcal{B}}\alpha\log(\pi_{\boldsymbol{\theta}_b}(\boldsymbol{a}_b|\boldsymbol{s}_b^t)) - Q_{\boldsymbol{\phi}}\big(\{\boldsymbol{s}_b^t\},\{\boldsymbol{a}_b\}\big)\Bigg]\Bigg]. \tag{32}$$

To update $\boldsymbol{\phi}$, we follow the centralized MADSAC step and minimize the following loss:

$$J_Q(\boldsymbol{\phi}) = \mathbb{E}_{\xi^t\sim\Xi}\left[\left(Q_{\boldsymbol{\phi}}\big(\{\boldsymbol{s}_b^t\},\{\boldsymbol{a}_b^t\}\big)-\big(r^{t+1}+\gamma\bar{V}(\{\boldsymbol{s}_b^{t+1}\})\big)\right)^2\right], \tag{33}$$

Algorithm 2 Decentralized Multi-Agent Discrete SAC-Based Cache Update

1: Initialize centralized soft Q-function parameters $\boldsymbol{\phi}_1, \boldsymbol{\phi}_2$, policy parameters $\{\boldsymbol{\theta}_b\}$
2: Initialize parameters of target networks $\boldsymbol{\phi}_i^- \leftarrow \boldsymbol{\phi}_i, i = 1, 2$
3: **for** $t = 0, 1, 2, \cdots$ **do**
4: Observe $\boldsymbol{s}_b$ and take action $\boldsymbol{a}_b \sim \pi_{\boldsymbol{\theta}_b}(\cdot|\boldsymbol{s}_b)$ for $b \in \mathcal{B}$
5: Observe $\boldsymbol{s}_b'$ for $b \in \mathcal{B}$ and reward r
6: Store $\xi = \big(\{\boldsymbol{s}_b\}, \{\boldsymbol{a}_b\}, r, \{\boldsymbol{s}_b'\}\big)$ in *RB*
7: **procedure** TRAINMADSAC
8: Randomly draw a batch of N experiences as Ξ_N
9: **for** each $\xi = \big(\{\boldsymbol{s}_b\}, \{\boldsymbol{a}_b\}, r, \{\boldsymbol{s}_b'\}\big) \in \Xi_N$ **do**
10: Calculate target values: $y_\xi = r + \gamma(\min_{i=1,2} Q_{\boldsymbol{\phi}_i^-}(\{\boldsymbol{s}_b'\}, \{\boldsymbol{a}_b'\}) - \alpha \sum_b \log \pi_{\boldsymbol{\theta}_b}(\boldsymbol{a}_b'|\boldsymbol{s}_b'))$, where $\boldsymbol{a}_b' = GS(\pi_{\boldsymbol{\theta}_b}(\cdot|\boldsymbol{s}_b'))$
11: **end for**
12: Update $\boldsymbol{\phi}_i$ by taking a one-step gradient descent of $\frac{1}{N}\sum_{\xi\in\Xi_N}(Q_{\boldsymbol{\phi}_i}(\{\boldsymbol{s}_b\}, \{\boldsymbol{a}_b\}) - y_\xi)^2$, for $i = 1, 2$
13: Update $\boldsymbol{\theta}_b$ by taking a one-step gradient descent of $\frac{1}{N}\sum_{\xi\in\Xi_N}\big(\min_{i=1,2} Q_{\boldsymbol{\phi}_i}(\{\boldsymbol{s}_b\}, \{GS(\pi_{\boldsymbol{\theta}_b}(\cdot|\boldsymbol{s}_b)\})) - \alpha \sum_b \log \pi_{\boldsymbol{\theta}_b}(GS(\pi_{\boldsymbol{\theta}_b}(\cdot|\boldsymbol{s}_b'))|\boldsymbol{s}_b')\big)$
14: Update temperature α by taking a one-step gradient descent of $-\frac{1}{N}\sum_{\xi\in\Xi_N}\alpha\sum_b\big(\log \pi_{\boldsymbol{\theta}_b}(GS(\pi_{\boldsymbol{\theta}_b}(\cdot|\boldsymbol{s}_b))|\boldsymbol{s}_b) - \bar{H}\big)$
15: Update parameters of target networks $\boldsymbol{\phi}_i^- \leftarrow \tau\boldsymbol{\phi}_i + (1-\tau)\boldsymbol{\phi}_i^-$, for $i = 1, 2$
16: **end procedure**
17: **end for**

where the target value $\bar{V}(\{\boldsymbol{s}^{t+1}\})$ is given by:

$$\mathbb{E}_{\boldsymbol{a}_b\sim\pi_{\boldsymbol{\theta}_b}}\left[Q_{\boldsymbol{\phi}^-}(\{\boldsymbol{s}^{t+1}\}, \{\boldsymbol{a}_b\}) - \alpha\sum_{b\in\mathcal{B}}\log\pi_{\boldsymbol{\theta}_b}(\boldsymbol{a}_b|\boldsymbol{s}_b^{t+1})\right], \tag{34}$$

and $Q_{\boldsymbol{\phi}^-}(\{\boldsymbol{s}_b\}, \{\boldsymbol{a}_b\})$ is the target network with parameter $\boldsymbol{\phi}^-$ being slowly updated every epoch. Notably, the target value in (34) is generally not equivalent to the target value in the centralized control, i.e., (28), because $\pi_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s}^{t+1}) \neq \sum_{b\in\mathcal{B}}\log\pi_{\boldsymbol{\theta}_b}(\boldsymbol{a}_b|\boldsymbol{s}_b^{t+1})$ usually holds. Finally, the temperature parameter α can be updated similarly to (29). Likewise, the expectation in (32)-(34) can be approximated by using samples drawn from the *RB* or produced by the GS-sampler. We present the algorithm implementation of MADSAC-decentralzed control (MADSAC-DC) in the ensuing subsection.

B. Algorithm Implementation

As illustrated in Fig. 2b, the proposed MADSAC-DC operates as follows: in the training procedure, each EN needs to upload local observations to the CP, which then globally optimizes policy parameters. Again, we adopt the technique of the clipped double Q-learning, and maintain two Q-networks with parameters $\boldsymbol{\phi}_i, i = 1, 2$. The training steps are summarized in Algorithm 2. Similarly, at every iteration, we draw a batch of experiences to estimate the expectation in loss functions, $J_\pi(\{\boldsymbol{\theta}_b\})$, $\{J_Q(\boldsymbol{\phi}_i)\}$ and $J(\alpha)$. Parameters $\boldsymbol{\phi}_1$ and $\boldsymbol{\phi}_2$ are updated in steps 11-14, whilst $\{\boldsymbol{\theta}_b\}$ and α are updated in step 15 and 16, respectively. Different from the centralized implementation, the decentralized policy for each EN is conditioned on its local observation instead of the state. Therefore, we have to sample every local action $\boldsymbol{a}_b$ from a decentralized policy, e.g., $GS(\pi_{\boldsymbol{\theta}_b}(\cdot|\boldsymbol{s}_b))$. Finally, the CP should deliver the well-trained policy parameters to each EN for online decision-making. At this stage, each EN can produce local actions based on local observation without introducing heavy fronthaul signaling overhead. Since all the decentralized policies are trained by the CP with the centralized soft-Q function, the computational complexity of MADSAC-DC is comparable to MADSAC-CC, e.g., grows quadratically with the number of ENs and linearly with the number of sensors.

Remark 2 *Moreover, we can generalize decentralized multi-agent SAC learning into a value decomposition-based diagram. Similar to conventional value decomposition-based approaches (e.g., the VDN, QMIX, and QTRAN in [39], [40]), the centralized soft Q-function can be factorized into individual soft Q-functions, which are used to fine-tune the decentralized policies for all ENs. This diagram provides an efficient way to train decentralized policies. Compared with the conventional approaches, the utilization of entropy regularization in the individual soft Q-function is envisioned to encourage exploration and improve performance.*

VI. PERFORMANCE EVALUATION

In this section, we evaluate the performance of the proposed algorithms under various kinds of scenarios. Unless otherwise stated, the default setting is as follows: three ENs are considered in an IoT sensing network; each EN has a communication range of 100 m and can coordinate 10 randomly distributed sensors; the storage of every content item is randomly generated within $[0.05, 0.1]$ GB. The range of the AoI is $[1, 50]$. Concerning communications between IoT sensors and ENs, the transmission power is 23 dBm at each sensor; the path loss is $-(148.1 + 37.6\log_{10} d)$ dB with d being the distance in km; the channel bandwidth is 10 MHz [7]; the antenna gain is 10 dBi; the log-normal shadowing parameter is 8 dB; the received SNR threshold is specified by $\eta_{th} = 10$ dB; and the parameter σ of the Rayleigh distribution equals 1. Furthermore, the space-time popularity dynamics of user requests are modeled as follows: user requests at distinct ENs exhibit individual content popularity distributions; at each EN, there are at most 100 users making requests according to a class of Zipf distributions [28], namely, $p_{f,b} = \zeta_{f,b}^{-\upsilon_b} / \sum_{f'\in\mathcal{F}}\zeta_{f',b}^{-\upsilon_b}, \forall f, b$, where υ_b denotes the skewness factor that is selected from $\{0.5, 1, 1.5, 2\}$, and $\{\zeta_{f,b}\}$ denote rank orders of content items that are dynamically evolving by following certain transition probability matrices [28]. In addition, we consider $\omega_1 = \omega_2 = 1$.

In the subsequent subsections, we consider the following algorithms for comparison:

- **DQN:** This algorithm is widely used in prior works (e.g., [12], [23]) to handle update at a single EN. It is expected to obtain near-optimal results in small-scale

settings, which attempts to validate the effectiveness of the proposed algorithm.

- **AC:** We consider a popular AC-based algorithm from [37]. To apply it in the discrete decision-making, similar to the proposed algorithms, we again adopt the GS-sampler and recast the output into low dimensional vectors. The objective of this approach is to evaluate the potential of entropy regularization in augmenting exploration.

The above-mentioned algorithms operate in the centralized manner, similar to MADSAC-CC.

A. Learning Curves of the Proposed Algorithms

We design either Q-networks or policy networks as five-layer neural networks, which consist of an input layer, three hidden layer, and an output layer, respectively. To stabilize training, the *polynomial learning rate* policy is adopted to train networks (readers are referred to [41] in detail.). We summarize key parameters of algorithm implementation in Table II. To ensure fairness, all algorithms are implemented by the same configuration.

We illustrate the learning curves of all the algorithms in Fig. 3a and 3b, respectively. Specifically, the algorithm performance is presented in terms of average reward (shown by learning curves) and standard deviation (shown by shaded areas). All of the results are obtained by applying the moving average, i.e., $\sum_{\tau=t-T+1}^{t} r^{\tau}$, where $T = 5000$. In Fig. 3a, we consider a single agent setting (i.e., EN) to validate the effectiveness of the proposed discrete variant of SAC. Clearly, the proposed MADSAC converges very fast and achieves comparable final results as that of DQN, whilst AC takes a much longer while to converge. Moreover, we can observe some sudden drops in the curve of AC, whereas the curve of the proposed one is generally flat. The observation indicates that the proposed approach is able to learn more stably. We further consider a multi-agent setting with three ENs in Fig. 3b. Evidently, the proposed algorithms outperform DQN and AC. At the initial stage, the rewards of MADSAC-CC and MADSAC-DC increase faster than AC, and shortly converge to almost the same level. We can observe notable gaps between final results of the proposed algorithms and AC. This finding implies that the entropy-regularized objective in the proposed approach is able to circumvent premature convergence somehow. However, DQN fails to make meaningful progress in the multi-agent setting. The reason is that an explosive action space makes it difficult for DQN to estimate the values of the Q-function. These results confirm the remarkable performance of the proposed algorithms in terms of convergence speed and final performance.

B. Scalability

In this subsection, we investigate the impacts of system parameters and study the scalability of the proposed algorithms. In what follows, we use the average weighted cost as the performance criterion, i.e., defined in (11). All of the results are obtained by averaging over 10000 epochs after DNNs are well-tuned.

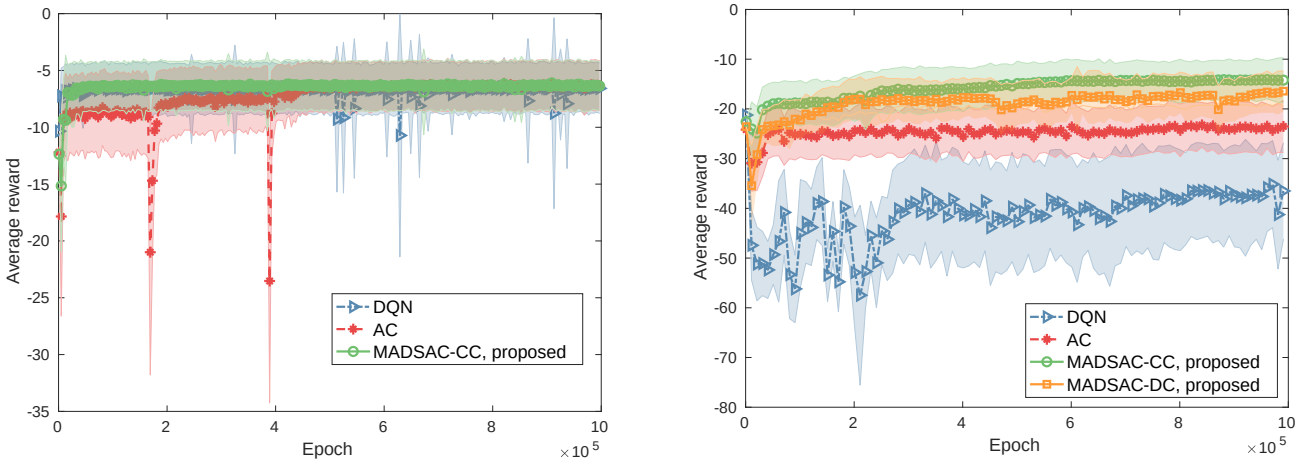
TABLE II: Parameters for algorithm implementations

Parameters	Value
Number of neurons in each hidden layer	128
Optimizer	*Adam*
Initial learning rate for Q-networks	0.01
Initial learning rate for policy networks	0.001
Power factor for decreasing learning rates	0.9
Memory capacity of *RB*	5000
Mini-batch size	100
Step size for updating target networks	0.001
Discount factor	0.99

Particularly, we first vary the number of agents (e.g., ENs) and plot the results in Fig. 4a. In general, the proposed decentralized algorithm performs close to the centralized one. This is because the use of a centralized critic in MADSAC-DC can effectively train local policies, which favorably augments EN collaboration and performance. Moreover, the proposed centralized and decentralized algorithms always obtain better results than the AC-based scheme. This phenomenon demonstrates the effectiveness of using an entropy regularizer to prevent premature convergence. As the number of agents becomes large, the performance of the proposed decentralized algorithm gradually degrades due to the restriction of local observations. It is worth mentioning that, when five or more ENs are considered, it is not practical to implement DQN due to the extremely large number of actions. Moreover, we illustrate the results of each considered metric (achieved by MADSAC-CC) in Fig. 4b. As can be seen, the average AoI becomes larger when more ENs are available. The reason is that more content items are involved with the growing number of ENs. The update costs (e.g., energy cost and traffic loads) also become larger as the number of ENs increases. However, when more than five ENs are installed, the update costs slightly decrease. Our conjecture is that, under a scenario with a large number of ENs, the summation of energy cost and traffic load may contribute to a much larger fraction of the objective function; thus, content items with rather small sizes and low transmission energy may be selected to balance the overall weighted cost.

To further investigate the scalability, we carry out experiments by changing the number of sensors within the communication range of each EN. As shown in Fig. 5a, the weighted cost, achieved by the decentralized design, is quite close to that of the centralized one. This observation further corroborates the remarkable performance of the decentralized control. As anticipated, MADSAC-CC and MADSAC-DC achieve much lower average weighted costs than DQN. The reason being that the conventional DQN needs to estimate the Q-function over all discrete state-action pairs, which causes the output size of the neural networks to be exponentially large. Specifically, when 20 sensors are deployed at the coverage of each EN, the proposed DRL schemes are able to reduce the weighted cost by 54.23%, 50.94%, respectively, compared with the DQN-based scheme. Notably, when 25 sensors are considered, the resulting number of discrete actions is 17576, making DQN implementation impossible. Similarly, the proposed DRL schemes outperform the AC-based scheme over the entire horizontal axis. However, we should mention

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCOMM.2021.3086535, IEEE Transactions on Communications



(a) Single agent

(b) Multiple agents.

Fig. 3: Learning curves.

that the utilization of entropy-regularization in the proposed algorithms does not alway have a remarkable advantage over the conventional RL. The AC based scheme sometimes achieves comparable performance as the proposed ones, e.g., in the case of 15 sensors. We further show the results of each performance criterion (achieved by MADSAC-CC) in Fig. 5b. The average AoI gradually becomes large because of the enlargement of content catalog. Average transmission energy and traffic loads exhibit similar results to what has been found in Fig. 4b.

The above-mentioned simulation results confirm the superiority of the proposed discrete variant of SAC, and the generalization of the decentralized approach. In the ensuing section, we only implement MADSAC-CC and focus on the tradeoff among the considered performance criteria.

C. Tradeoff among AoI & Energy Consumption & Traffic Loads

In this subsection, we investigate the tradeoff among average AoI, transmission energy consumption and fronthaul traffic loads. To benchmark how energy consumption and traffic load consideration compromises the performance of data freshness, we consider two schemes as follows. i) *Age-Optimal Scheme:* We only optimize the average AoI without incorporating cache update costs; ii) *Random Scheme:* at each epoch, we randomly update one content item at each EN without being aware of the tradeoff among the considered performance criteria. The proposed scheme is referred to as *cost-effective scheme*.

We first carry out experiments by varying ω_1 and illustrate the results of average AoI, transmission energy consumption, and fronthaul traffic loads in Fig. 6 - 8, respectively. Besides, ω_2 is fixed as the default value (i.e., 1). Evidently, when we enlarge the weight for transmission energy (i.e., ω_1), the average AoI continuously grows high whereas transmission energy drops off quickly. This finding implies that the cost-effective scheme attempts to reduce the frequency of content update. For this reason, it leads to the reduction of traffic loads simultaneously, although the associated weight is fixed as a constant.

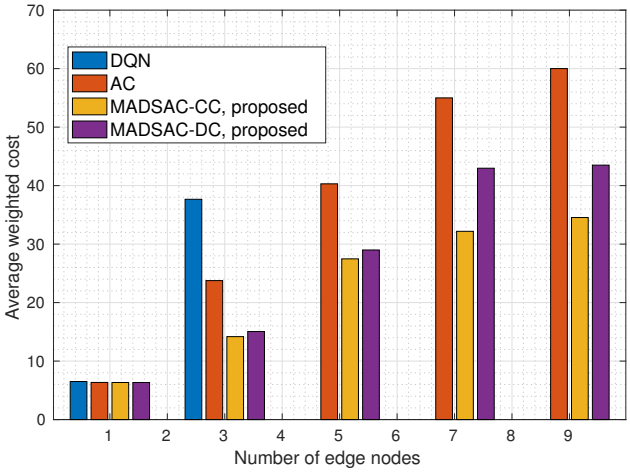
We then conduct simulations by changing ω_2 and fixing $\omega_1 = 1$. The results are depicted in Fig. 9 - Fig. 11. It can be observed that, with the increment of ω_2, fronthaul traffic loads decrease gradually while the average AoI becomes increasingly large. Similarly, we conjecture that the update of the content becomes less frequent, which somehow results in more outdated content items cached at ENs. In addition, it should be noted that fronthaul traffic loads, achieved by enlarging ω_2 (shown in Fig. 11), decreases faster than that in Fig. 8. For instance, when increasing ω_2 up to 10, there is a 67.92% reduction of traffic loads, which is much larger than 42.16% achieved by enlarging ω_1. This is because, when we enlarge ω_2, the cost-effective scheme is likely to reduce the update frequencies of content items having large storage size. A similar conclusion can be drawn towards the degradation of energy consumption by tuning ω_1. When we set ω_1 or ω_2 to be larger than 10, the reduction of energy consumption and fronthaul traffic loads are quite limited in comparison to age-optimal scheme or random scheme; it however leads to much larger AoI compared with baselines.

Finally, we illustrate the impact of the number of updates at each EN on Fig. 12. As can be observed, with the increase in the number of updates per epoch, the average AoI decreases gradually whilst the update costs, e.g., average energy consumption and traffic load, become larger. Note that when we perform three updates or more, the average AoI decreases slowly. This is because some updated content items may receive low attention from users under these circumstances. Therefore, it is likely that updating unpopular content may reduce the average AoI marginally yet at the cost of much more update costs. In practical system designs with energy provision and fronthaul capacity constraints, we need to reasonably select the number of updates to confine the update cost whilst minimizing the average AoI.

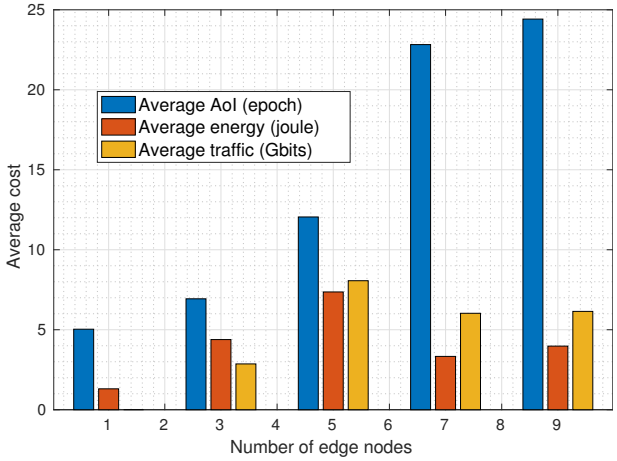
VII. Conclusion

In this paper, we have developed a multi-agent reinforcement learning framework for cache update in IoT sensing

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCOMM.2021.3086535, IEEE Transactions on Communications

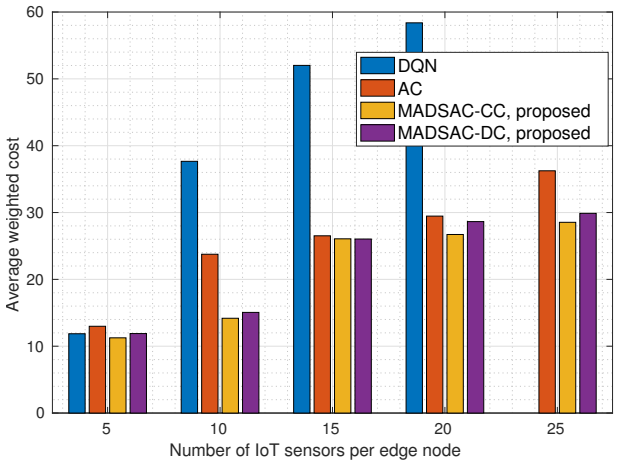


(a) Average weighted cost.

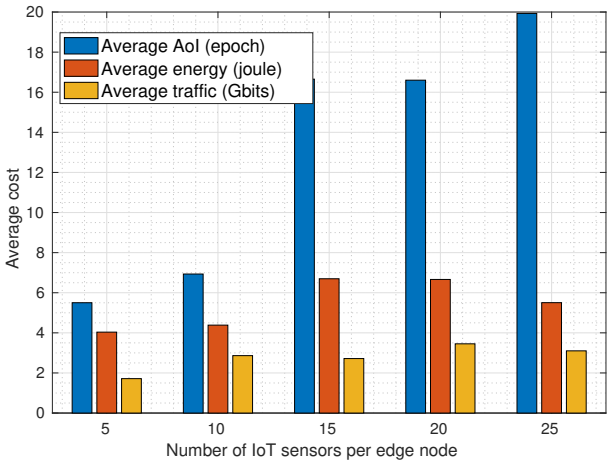


(b) Individual performance criterion.

Fig. 4: Impact of the number of ENs.



(a) Average weighted cost.



(b) Individual performance criterion.

Fig. 5: Impact of the number of IoT sensors at each EN.

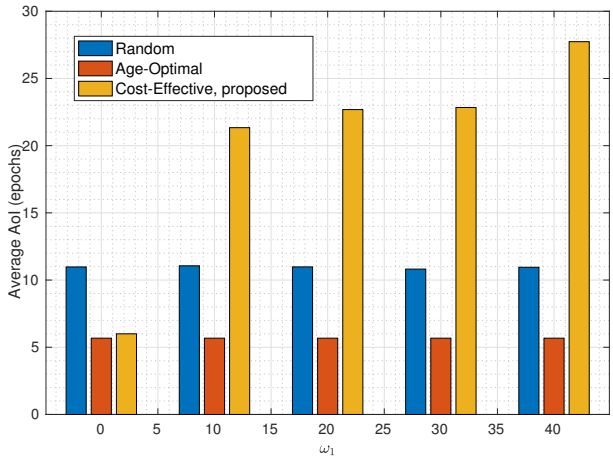


Fig. 6: Average AoI versus the energy factor ω_1.

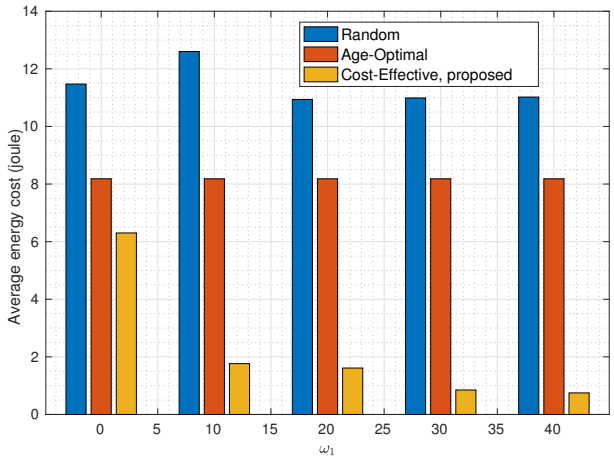


Fig. 7: Average transmission energy consumption versus the energy factor ω_1.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCOMM.2021.3086535, IEEE Transactions on Communications

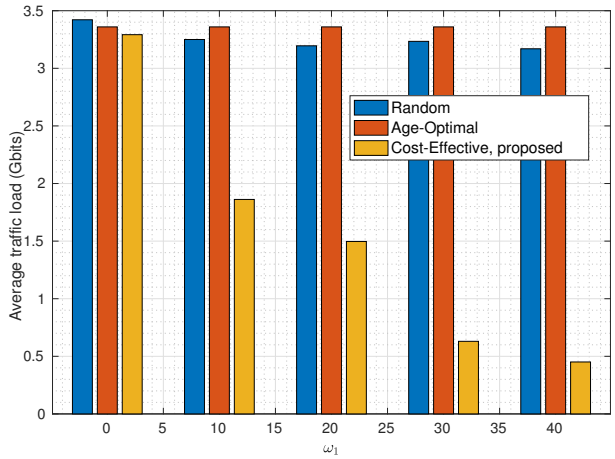


Fig. 8: Average fronthaul traffic loads versus the energy factor ω_1.

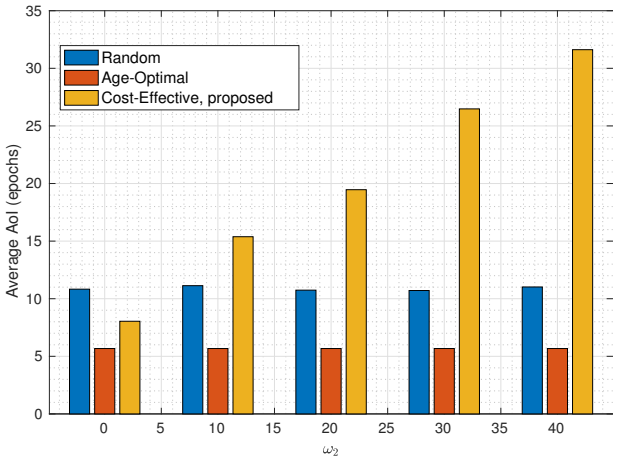


Fig. 9: Average AoI versus the traffic factor ω_2.

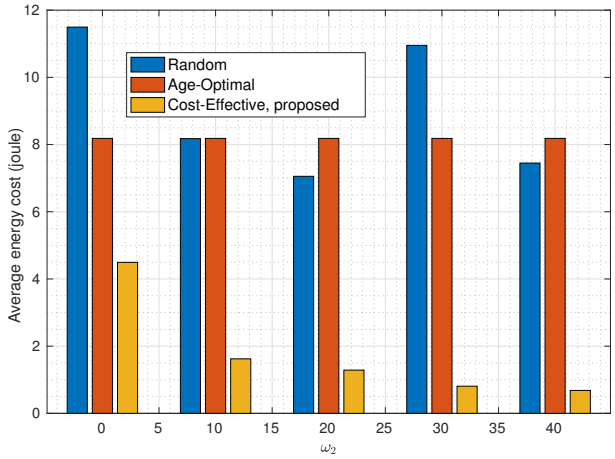


Fig. 10: Average transmission energy consumption versus the traffic factor ω_2.

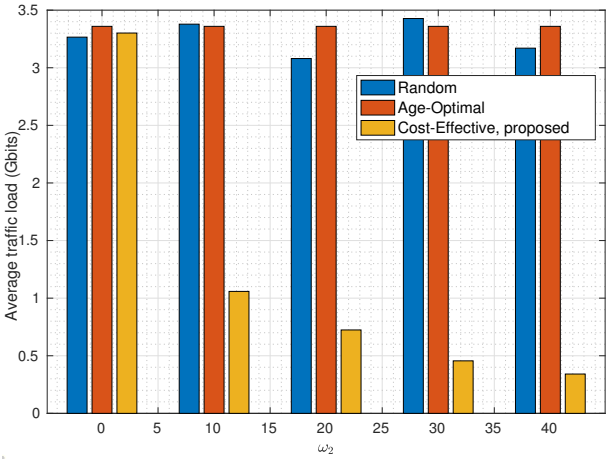


Fig. 11: Average fronthaul traffic loads versus the traffic factor ω_2.

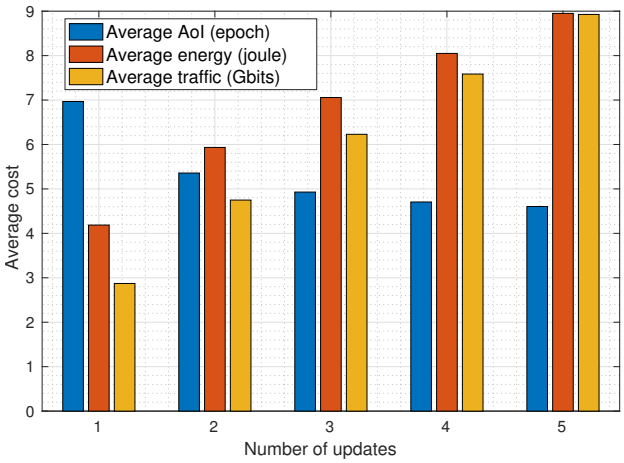


Fig. 12: Impact of the number of updates at each EN per epoch.

networks. The objective of this framework is to minimize the weighted average age of information plus energy cost as well as fronthaul traffic loads. We have derived a characterization of energy consumption for content delivery. To cope with the discrete multi-agent decision-making, we have proposed a novel reinforcement learning approach with low space complexity. Simulation results have indicated that the proposed algorithms significantly outperform deep Q-network and traditional actor-critic approaches as the number of edge nodes or sensors increases; and the proposed decentralized caching scheme obtains satisfactory performance compared with the centralized one. The developed approach also has great potential to address many other multi-agent discrete decision-making problems in communication and networking.

APPENDIX

A. *Proof of Proposition 1*

Given $\mathbb{P}_{\kappa_f}(\kappa_f) = \frac{2\kappa_f}{\sigma}\exp(-\frac{\kappa_f^2}{\sigma})$, the squared envelope of the small scale fading κ_f^2 follows an exponential distribution, $\frac{1}{\sigma}\exp(-\frac{\kappa_f^2}{\sigma})$. Therefore, the distribution of the received SNR

η_f is given by $\mathbb{P}(\eta_f) = \frac{1}{2\beta_f}\exp(-\frac{\eta_f}{2\beta_f})$. Recall that content transmissions are effective only when the received SNR exceeds η_{th}. As a result, the expected throughput $\bar{R}_f$ can be calculated as follows:

$$\begin{aligned}\bar{R}_f &= \int_{\eta_{th}}^{\infty} B_0 \log_2(1+\eta_f)\mathbb{P}(\eta_f)\ d\eta_f \\ &= \frac{B_0}{\ln 2}\int_{\eta_{th}}^{\infty} \log(1+\eta_f)\frac{1}{2\beta_f}\exp\left(-\frac{\eta_f}{2\beta_f}\right)\ d\eta_f \\ &= -\frac{B_0}{\ln 2}\int_{\eta_{th}}^{\infty} \log(1+\eta_f)\ d\exp\left(-\frac{\eta_f}{2\beta_f}\right) \\ &= R_{th}\exp\left(-\frac{\eta_f}{2\beta_f}\right) + \frac{B_0}{\ln 2}\int_{\eta_{th}+1}^{\infty}\exp\left(\frac{1-\eta_f}{2\beta_f}\right)\frac{1}{\eta_f}\ d\eta_f \\ &= R_{th}\exp\left(-\frac{\eta_f}{2\beta_f}\right) + \frac{B_0}{\ln 2}\exp\left(\frac{1}{2\beta_f}\right)\rho_f(\eta_{th}+1),\end{aligned}$$

where function $\rho_f(\cdot)$ is defined by (5). Given the content size s_f and transmission power P_f, the average energy consumption can be given by $\bar{E}_f = P_f s_f/\bar{R}_f$. This completes the proof.

B. Proof of Lemma 3

For notational convenience, we denote $\alpha_i = \mu_{\boldsymbol{\theta}}(i|\boldsymbol{s})$, and $\varpi_i = g_{i,b} + \log\alpha_i, \forall i \in \mathcal{A}_b, b \in \mathcal{B}$. Then, it follows that $\hat{\boldsymbol{a}}_b = \arg\max_{i\in\mathcal{A}_b}\varpi_i$. We calculate the following probability $\Pr\{\hat{\boldsymbol{a}}_b = i|\boldsymbol{s}\} = \Pr\{\varpi_i \geq \varpi_j, \forall j \neq i\}$:

$$\begin{aligned}&= \int_{-\infty}^{\infty}\Pi_{j\neq i}\{\varpi_i \geq \varpi_j|\varpi_i\}\Pr\{\varpi_i\}d\varpi_i \\ &= \int_{-\infty}^{\infty}\Pi_{j\neq i}\exp\left(-\exp\left(-\varpi_i+\ln\alpha_j\right)\right) \\ &\qquad \exp\left(-(\varpi_i - \ln\alpha_j + \exp\left(-(\varpi_i - \ln(\alpha_i))\right))\right)d\varpi_i \\ &= \int_{-\infty}^{\infty}\exp(-\Sigma_{j\neq i}\alpha_j\exp(-\varpi_i)) \\ &\qquad \alpha_i\exp\left(-(\varpi_i + \alpha_i\exp(-\varpi_i))\right)d\varpi_i \\ &\overset{(a)}{=} \int_{-\infty}^{\infty}\alpha_i\exp\left(-\varpi_i - \exp(-\varpi_i)\right)d\varpi_i \\ &= \alpha_i\end{aligned}$$

for $\forall i \in \mathcal{A}_b, b \in \mathcal{B}$, where step (a) is a result of $\sum_{i\in\mathcal{A}_b}\mu_{\boldsymbol{\theta}}(i|\boldsymbol{s}) = 1$. Then, we conclude that $\Pr\{\hat{\boldsymbol{a}}|\boldsymbol{s}\} = \Pi_{b\in\mathcal{B}}\Pr\{\hat{\boldsymbol{a}}_b|\boldsymbol{s}\} = \Pi_{b\in\mathcal{B}}\,\mu_{\boldsymbol{\theta}}(\hat{\boldsymbol{a}}_b|\boldsymbol{s})$, which completes the proof.

REFERENCES

[1] S. Madakam, R. Ramaswamy, and S. Tripathi, "Internet of things (IoT): A literature review," *Int. J. Comput. Commun.*, vol. 3, no. 5, pp. 164 – 173, May 2015.

[2] Y. He, F. R. Yu, N. Zhao, V. C. Leung, and H. Yin, "Software-defined networks with mobile edge computing and caching for smart cities: A big data deep reinforcement learning approach," *IEEE Commun. Mag.*, vol. 55, no. 12, pp. 31–37, Dec. 2017.

[3] M. Chen, U. Challita, W. Saad, C. Yin, and M. Debbah, "Artificial neural networks-based machine learning for wireless networks: A tutorial," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 4, pp. 3039–3071, Fourthquarter 2019.

[4] T. X. Tran, A. Hajisami, P. Pandey, and D. Pompili, "Collaborative mobile edge computing in 5G networks: New paradigms, scenarios, and challenges," *IEEE Commun. Mag.*, vol. 55, no. 4, pp. 54–61, Apr. 2017.

[5] L. Wang, H. Wu, Z. Han, P. Zhang, and H. V. Poor, "Multi-hop cooperative caching in social IoT using matching theory," *IEEE Trans. Wireless Commun.*, vol. 17, no. 4, pp. 2127–2145, Apr. 2018.

[6] X. Wu, Q. Li, X. Li, V. C. Leung, and P. Ching, "Joint long-term cache updating and short-term content delivery in cloud-based small cell networks," *IEEE Trans. Commun.*, vol. 68, no. 5, pp. 3173–3186, May 2020.

[7] X. Wu, Q. Li, V. C. Leung, and P. Ching, "Joint fronthaul multicast and cooperative beamforming for cache-enabled cloud-based small cell networks: An MDS codes-aided approach," *IEEE Trans. Wireless Commun.*, vol. 18, no. 10, pp. 4970–4982, Oct. 2019.

[8] X. Li, X. Wang, P.-J. Wan, Z. Han, and V. C. Leung, "Hierarchical edge caching in device-to-device aided mobile networks: Modeling, optimization, and design," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 8, pp. 1768–1785, June 2018.

[9] J. Li, Y. Chen, Z. Lin, W. Chen, B. Vucetic, and L. Hanzo, "Distributed caching for data dissemination in the downlink of heterogeneous networks," *IEEE Trans. Commun.*, vol. 63, no. 10, pp. 3553–3568, Oct. 2015.

[10] J. Li, H. Chen, Y. Chen, Z. Lin, B. Vucetic, and L. Hanzo, "Pricing and resource allocation via game theory for a small-cell video caching system," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 8, pp. 2115–2129, Aug., 2016.

[11] X. Wu, J. Li, M. Xiao, P. Ching, and H. V. Poor, "Multi-agent reinforcement learning for cooperative coded caching via homotopy optimization," *IEEE Trans. Wireless Commun.*, to appear, 2021.

[12] C. Xu, X. Wang, H. H. Yang, H. Sun, and T. Q. Quek, "AoI and energy consumption oriented dynamic status updating in caching enabled IoT networks," in *Proc. IEEE Conf. Comput. Commun. Workshop*, July 2020, pp. 710–715.

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

[14] L. Huang and E. Modiano, "Optimizing age-of-information in a multi-class queueing system," in *Proc. IEEE Int. Symp. Inf. Theory*, June 2015, pp. 1681–1685.

[15] M. Costa, M. Codreanu, and A. Ephremides, "On the age of information in status update systems with packet management," *IEEE Trans. Inf. Theory*, vol. 62, no. 4, pp. 1897–1910, Apr. 2016.

[16] 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.

[17] M. Emara, H. ElSawy, and G. Bauch, "A spatiotemporal framework for information freshness in IoT uplink networks," *arXiv preprint arXiv:2001.11333*, 2020.

[18] B. Zhou and W. Saad, "Joint status sampling and updating for minimizing age of information in the internet of things," *IEEE Trans. Commun.*, vol. 67, no. 11, pp. 7468–7482, Nov. 2019.

[19] 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.*, Sept. 2018, pp. 1967–1971.

[20] M. Ma and V. W. Wong, "A deep reinforcement learning approach for dynamic contents caching in HetNets," in *Proc. IEEE Int. Conf. Commun.*, June 2020, pp. 1–6.

[21] E. T. Ceran, D. Gündüz, and A. György, "Reinforcement learning to minimize age of information with an energy harvesting sensor with HARQ and sensing cost," in *Proc. IEEE Conf. Comput. Commun. Wkshps*, Apr. 2019, pp. 656–661.

[22] M. Hatami, M. Jahandideh, M. Leinonen, and M. Codreanu, "Age-aware status update control for energy harvesting IoT sensors via reinforcement learning," in *Proc. IEEE Int. Symp. Pers. Indoor Mobile Radio Commun.*, Aug. 2020, pp. 1–6.

[23] M. A. Abd-Elmagid, H. S. Dhillon, and N. Pappas, "A reinforcement learning framework for optimizing age of information in RF-powered communication systems," *IEEE Trans. Commun.*, vol. 68, no. 8, pp. 4747 – 4760, Aug., 2020.

[24] H. Zhu, Y. Cao, X. Wei, W. Wang, T. Jiang, and S. Jin, "Caching transient data for internet of things: A deep reinforcement learning approach," *IEEE Internet Things J.*, pp. 2074 – 2083, Apr. 2019.

[25] J. Yao and N. Ansari, "Caching in dynamic IoT networks by deep reinforcement learning," *IEEE Internet Things J.*, to appear, 2020.

[26] X. Wu, X. Li, J. Li, P. Ching, and H. V. Poor, "Deep reinforcement learning for IoT networks: Age of information and energy cost tradeoff," in *Proc. IEEE Global Commun. Conf.*, Dec. 2020.

[27] M. Hassanalieragh, A. Page, T. Soyata, G. Sharma, M. Aktas, G. Mateos, B. Kantarci, and S. Andreescu, "Health monitoring and management using internet-of-things (IoT) sensing with cloud-based processing: Opportunities and challenges," in *Proc. IEEE SCC*, June 2015, pp. 285–292.

[28] A. Sadeghi, F. Sheikholeslami, and G. B. Giannakis, "Optimal and scalable caching for 5G using reinforcement learning of space-time popularities," *IEEE J. Sel. Topics Signal Process.*, vol. 12, no. 1, pp. 180–190, Feb. 2018.
[29] K. Zhang, Z. Yang, and T. Başar, "Multi-agent reinforcement learning: A selective overview of theories and algorithms," *arXiv preprint arXiv:1911.10635*, 2019.
[30] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel *et al.*, "Soft actor-critic algorithms and applications," *arXiv preprint arXiv:1812.05905*, 2018.
[31] E. Gindullina, L. Badia, and D. Gündüz, "Age-of-information with information source diversity in an energy harvesting system," *arXiv preprint arXiv:2004.11135*, 2020.
[32] G. Mountaser, M. L. Rosas, T. Mahmoodi, and M. Dohler, "On the feasibility of MAC and PHY split in cloud RAN," in *Proc. IEEE Wireless Commun. Netw. Conf.*, Mar. 2017, pp. 1–6.
[33] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, MIT Press, 2018.
[34] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, Feb. 2015.
[35] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with A stochastic actor," *arXiv preprint arXiv:1801.01290*, 2018.
[36] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with Gumbel-Softmax," *arXiv preprint arXiv:1611.01144*, 2016.
[37] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. M. O. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *CoRR*, vol. abs/1509.02971, 2015.
[38] S. Fujimoto, H. Van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," *arXiv preprint arXiv:1802.09477*, 2018.
[39] T. Rashid, M. Samvelyan, C. S. De Witt, G. Farquhar, J. Foerster, and S. Whiteson, "QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning," *arXiv preprint arXiv:1803.11485*, 2018.
[40] K. Son, D. Kim, W. J. Kang, D. E. Hostallero, and Y. Yi, "Qtran: learning to factorize with transformation for cooperative multi-agent reinforcement learning," *arXiv preprint arXiv:1905.05408*, 2019.
[41] P. Mishra and K. Sarawadekar, "Polynomial learning rate policy with warm restart for deep neural network," in *Proc. IEEE Region 10 Conf.*, Oct. 2019, pp. 2087–2092.

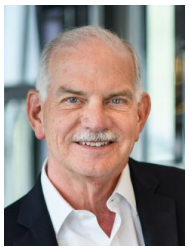
Xiongwei Wu received the B.Eng. in Electronic Information Engineering from University of Electronic Science and Technology of China, Chengdu, China, in 2016. He is currently working toward the Ph.D. degree with the Chinese University of Hong Kong (CUHK), Shatin, Hong Kong SAR, China. From August 2018 to December 2018, he was a Visiting International Research Student with the University of British Columbia (UBC), Vancouver, BC, Canada. From July 2019 to January 2020, he was a Visiting Student Research Collaborator with Princeton University, Princeton, NJ, USA. His research interests include signal processing and resource allocation in wireless networks, decentralized optimization, and machine learning.

Xiuhua Li received the B.S. degree from the Honors School, Harbin Institute of Technology, Harbin, China, in 2011, the M.S. degree from the School of Electronics and Information Engineering, Harbin Institute of Technology, in 2013, and the Ph.D. degree from the Department of Electrical and Computer Engineering, The University of British Columbia, Vancouver, BC, Canada, in 2018. He joined Chongqing University through One-Hundred Talents Plan of Chongqing University in 2019. He is currently a tenure-track Assistant Professor with the School of Big Data & Software Engineering, and the Dean of the Institute of Intelligent Software and Services Computing associated with Key Laboratory of Dependable Service Computing in Cyber Physical Society, Chongqing University, Chongqing, China. His current research interests are 5G/B5G mobile Internet, mobile edge computing and caching, big data analytics and machine learning.

Jun Li received Ph. D degree in Electronic Engineering from Shanghai Jiao Tong University, Shanghai, P. R. China in 2009. From January 2009 to June 2009, he worked in the Department of Research and Innovation, Alcatel Lucent Shanghai Bell as a Research Scientist. From June 2009 to April 2012, he was a Postdoctoral Fellow at the School of Electrical Engineering and Telecommunications, the University of New South Wales, Australia. From April 2012 to June 2015, he was a Research Fellow at the School of Electrical Engineering, the University of Sydney, Australia. From June 2015 to now, he is a Professor at the School of Electronic and Optical Engineering, Nanjing University of Science and Technology, Nanjing, China. He was a visiting professor at Princeton University from 2018 to 2019. His research interests include network information theory, game theory, distributed intelligence, multiple agent reinforcement learning, and their applications in ultra-dense wireless networks, mobile edge computing, network privacy and security, and industrial Internet of things. He has co-authored more than 200 papers in IEEE journals and conferences, and holds 1 US patents and more than 10 Chinese patents in these areas. He was serving as an editor of IEEE Communication Letters and TPC member for several flagship IEEE conferences. He received Exemplary Reviewer of IEEE Transactions on Communications in 2018, and best paper award from IEEE International Conference on 5G for Future Wireless Networks in 2017.

P. C. Ching received the B. Eng. (1st Class Honors) and Ph.D. degrees from the University of Liverpool, UK, in 1977 and 1981 respectively. From 1981 to 1982, he was Research Officer at the University of Bath, UK. In 1982, Prof. Ching returned to Hong Kong and joined the then Hong Kong Polytechnic as a lecturer. Since 1984, he has been with the Department of Electronic Engineering of the Chinese University of Hong Kong (CUHK), where he is currently Choh-Ming Li Professor of Electronic Engineering. He was Department Chairman from 1995 to 1997, Dean of Engineering from 1997 to 2003 and Head of Shaw College from 2004 to 2008. He became Director of the Shun Hing Institute of Advanced Engineering in 2004. From 2006 till end of 2014, Prof Ching was appointed as Pro-Vice-Chancellor/Vice-President of CUHK. Between 2013 to 2014, Prof. Ching also took up the Directorship of the CUHK Shenzhen Research Institute. Prof. Ching is very active in promoting professional activities, both in Hong Kong and overseas. He was a council member of the Institution of Electrical Engineers (IEE), past chairman of the IEEE Hong Kong Section, an associate editor of the IEEE Transactions on Signal Processing from 1997 to 2000 and IEEE Signal Processing Letters from 2001 to 2003. He was also a member of the Technical Committee of the IEEE Signal Processing Society from 1996 to 2004. He was appointed Editor-in-Chief of the HKIE Transactions between 2001 and 2004. He has been an Honorary Member of the editorial committee for Journal of Data Acquisition and Processing since 2000. Prof. Ching has been instrumental in organizing many international conferences in Hong Kong including the 1997 IEEE International Symposium on Circuits and Systems where he was the Vice-Chairman. He also served as Technical Program Co-Chair of the 2003 and 2016 IEEE International Conference on Acoustics, Speech and Signal Processing. Prof Ching was awarded the IEEE Third Millennium Award in 2000 and the HKIE Hall of Fame in 2010. In addition, Prof. Ching also plays an active role in community services. He was awarded the Silver Bauhinia Star (SBS) and the Bronze Bauhinia Star (BBS) by the HKSAR Government in 2017 and 2010, respectively, in recognition of his long and distinguished public and community services. He is presently Chairman of the Board of Directors of the Nano and Advanced Materials Institute Limited, Council Member of the Shaw Prize Foundation, as well as a Member of the Museum Advisory Committee (MAC) and the Chairperson of its Science Sub-committee. He is elected as President of the Hong Kong Academy of Engineering Sciences (HKAES) in 2018.

Victor C. M. Leung is a Distinguished Professor of Computer Science and Software Engineering at Shenzhen University, China. He is also an Emeritus Professor of Electrical and Computer Engineering and Director of the Laboratory for Wireless Networks and Mobile Systems at the University of British Columbia (UBC), Canada. His research is in the broad areas of wireless networks and mobile systems, and he has published widely in these areas. Dr. Leung is serving on the editorial boards of the IEEE Transactions on Green Communications and Networking, IEEE Transactions on Cloud Computing, IEEE Access, IEEE Network, and several other journals. He received the 1977 APEBC Gold Medal, 1977-1981 NSERC Postgraduate Scholarships, IEEE Vancouver Section Centennial Award, 2011 UBC Killam Research Prize, 2017 Canadian Award for Telecommunications Research, 2018 IEEE TCGCC Distinguished Technical Achievement Recognition Award, and 2018 ACM MSWiM Reginald Fessenden Award. He co-authored papers that won the 2017 IEEE ComSoc Fred W. Ellersick Prize, 2017 IEEE Systems Journal Best Paper Award, 2018 IEEE CSIM Best Journal Paper Award, and 2019 IEEE TCGCC Best Journal Paper Award. He is a Life Fellow of IEEE, and a Fellow of the Royal Society of Canada (Academy of Science), Canadian Academy of Engineering, and Engineering Institute of Canada. He is named in the current Clarivate Analytics list of "Highly Cited Researchers".



H. Vincent Poor received the Ph.D. degree in EECS from Princeton University in 1977. From 1977 until 1990, he was on the faculty of the University of Illinois at Urbana-Champaign. Since 1990 he has been on the faculty at Princeton, where he is currently the Michael Henry Strater University Professor. During 2006 to 2016, he served as the dean of Princetons School of Engineering and Applied Science. He has also held visiting appointments at several other universities, including most recently at Berkeley and Cambridge. His research interests are in the areas of information theory, machine learning and network science, and their applications in wireless networks, energy systems and related fields. Among his publications in these areas is the forthcoming book *Machine Learning and Wireless Communications*. (Cambridge University Press, 2021).

Dr. Poor is a member of the National Academy of Engineering and the National Academy of Sciences and is a foreign member of the Chinese Academy of Sciences, the Royal Society, and other national and international academies. Recent recognition of his work includes the 2017 IEEE Alexander Graham Bell Medal and a D.Eng. honoris causa from the University of Waterloo awarded in 2019.