Age of Information Driven Cache Content Update Scheduling for Dynamic Contents in Heterogeneous Networks

Manyou Ma, *Student Member, IEEE* and Vincent W.S. Wong *Fellow, IEEE*

***Abstract*—The recent development in mobile edge computing necessitates caching of dynamic contents, where new versions of contents become available around-the-clock, thus timely update is required to ensure their relevance. The age of information (AoI) is a performance metric that evaluates the freshness of contents. Existing works on AoI-optimization of cache content update algorithms focus on minimizing the long-term average AoI of all cached contents. Sometimes, user requests that need to be served in the future are known in advance and can be stored in user request queues. In this paper, we propose dynamic cache content update scheduling algorithms that exploit the user request queues. We consider a use case, where the trained neural networks (NNs) from deep learning models are being cached in a heterogeneous network (HetNet), as a motivating example. A queue-aware cache content update scheduling algorithm based on constrained Markov decision process (CMDP) is developed to minimize the average AoI of the dynamic contents delivered to the users. By using enforced decomposition technique and deep reinforcement learning, we propose two low-complexity suboptimal scheduling algorithms. Simulation results show that our proposed algorithms outperform the periodic cache content update scheme and reduce the average AoI by up to 30%.**

***Index Terms*—Age of information (AoI), dynamic content caching, constrained Markov decision process (CMDP), deep reinforcement learning (DRL) heterogeneous network (HetNet), queue-aware scheduling**

I. Introduction

To handle the ever-increasing growth of data traffic, one promising approach is to cache popular contents using a heterogeneous network (HetNet) architecture. In HetNet caching [2], a macro base station (MBS) and multiple small-cell base stations jointly serve users within a macrocell. These small-cell base stations have storage capacity and can act as content servers (CSs). When the data traffic is low, such as during off-peak hours, the MBS pushes popular contents to the CSs via a wireless backhaul. Henceforth, the CSs can simultaneously serve mobile user requests using their cached contents, resulting in lower power consumption and higher throughput. Previous research has studied different aspects of content caching, such as predicting future content popularity [3], [4], content placement strategies [5], [6], and scheduling algorithms design [7]–[10]. In the aforementioned works, the content caching strategies of *static files*, such as popular video and audio files, are studied. These files typically do not change once they have been created and hence only need to be pushed to the CSs once.

However, with the proliferation of Internet of things (IoT) and mobile edge computing paradigm, billions of IoT devices, *e.g.*, industrial robots, security cameras, and autonomous driving cars, are expected to be connected to the fifth generation (5G) and beyond wireless networks [11]. Many of these IoT applications require to download the latest version of *dynamic contents*, such as up-to-date software, list of neighbouring devices, environmental parameters (*e.g.*, temperature, traffic status, video feeds), and ledgers for blockchain-enabled applications [12], in order to perform either mission-critical or time-sensitive tasks using the onboard chipset. These contents are *dynamic* because newer versions of such contents become available around-the-clock. Ensuring the freshness of the dynamic contents that are delivered to the IoT applications is of equal importance as satisfying the conventional quality-of-service (QoS) requirements, such as average delay and throughput.

As an example, a particular genre of artificial intelligence (AI)-oriented IoT applications is powered by deep learning (DL) algorithms [13]. DL techniques have been applied ubiquitously in domains such as autonomous driving, natural language processing, and medical diagnosis. Since the training of DL neural networks (NNs) is computationally and memory intensive, general-purpose cloud computing facilities, such as the Amazon Web Service (AWS) and Microsoft Azure platforms, have been developed to train and maintain NNs, using an ever-growing training dataset with new data continuously added into those platforms. Once an NN has been trained, its size is typically small[1] compared to the raw data (*e.g.*, images, videos) collected by the IoT devices. Moreover, executing an NN in the deployment stage is less resource-demanding, compared to the training step in DL. Therefore, it is desirable for the IoT devices to download the trained NNs and execute the AI applications using their onboard

Manuscript received on Oct. 6, 2019; revised on Apr. 13, 2020 and Jun. 24, 2020; accepted on Sept. 4, 2020.

This work was supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC).

This paper has been published in part in the *Proceedings of the IEEE International Conference on Communications (ICC)*, Jun. 2020 [1]. The editor coordinating the review of this paper and approving it for publication was Besma Smida. *(Corresponding author: Vincent W.S. Wong)*

The authors are with the Department of Electrical and Computer Engineering, The University of British Columbia, Vancouver, BC, V6T 1Z4, Canada (e-mail: {manyoum, vincentw}@ece.ubc.ca).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

[1]The size of popular pre-trained NNs ranges from 5 MB (SqueezeNet) to 500 MB (VGG11) [14].

chipset. Tools have been developed for the deployment of DL algorithms on lightweight computational devices, such as smartphones [15]. In the literature, the DL frameworks, where NNs are trained in a centralized server and later distributed to the users in the system, have already been proposed for wireless communication and robotics applications [16], [17]. In anticipation of the ubiquitous adoption of these DL frameworks, effective algorithms need to be developed to deliver the trained NNs to the system users. We postulate that the trained NNs should be treated as *dynamic contents*, since we live in a dynamically changing world with the explosive emergence of new information, patterns, and trends, the DL NNs need to be re-trained using the newly available data to stay adaptive to these new changes. Due to the massive number of IoT devices connected to the networks, many of which with stringent latency requirements, it may not be possible for the IoT devices to download the dynamic contents, such as the NNs, directly from the cloud computing server. This is because sending all these data packets (with the dynamic contents as payloads) across the cloud through the core network to the radio access network introduces extra delay overhead, and may increase the level of congestion in the core network and the access links. Hence, the aforementioned HetNet architecture can be adopted to tackle the dynamic contents caching problem. To reduce the data traffic in the MBS, recent versions of the dynamic contents can be cached in the CSs. When a user request arrives, the cached dynamic content can be sent by the CS to the user at the *target download time* specified by the user.

To ensure the freshness of the dynamic contents delivered to the IoT applications, we propose to use the *age of information* (AoI) [18], [19] of the delivered dynamic contents as a metric to evaluate the system performance. The AoI of a file depicts the amount of time that has elapsed since the current version of a file is generated. Hence, a smaller AoI corresponds to a file which is more recent. Modeling and optimizing the AoI of a system have attracted much research interest. For example, in the original work that proposed the AoI framework, Kaul *et al.* in [18] studied the optimal rate of re-sampling different measurement data in the control system of a vehicle, using different queuing models of user arrivals. In [20], Kadota *et al.* studied the problem of optimizing the AoI in a wireless sensor network subject to the throughput constraint. In [21]–[23], scheduling policies for wireless broadcasting channels are proposed. In [24], Yates *et al.* used AoI as a metric to evaluate the performance of a caching network. Following [24], research on dynamic content caching [25], [26] has also been reported in the literature. Although the earlier works on AoI optimization are based on queuing theory frameworks, learning-based methods such as Markov decision process (MDP) [27]–[29] and reinforcement learning (RL) [30] have also been proposed recently.

In the aforementioned works on AoI optimization, stochastic arrivals of user requests following a renewal process are assumed, and the long-term average AoI of all the files in the system is minimized. However, we conjecture that in practical systems, many user requests may require the dynamic content to be sent at a specific time in future, because IoT devices in general submit their request earlier than the expected time that the dynamic content is being used. Therefore, the number of dynamic contents or files that need to be transmitted in the near future are often known ahead of time and can be used to facilitate scheduling of dynamic content update. In this paper, we consider the scenario where user requests arrive before their *target download time*. We employ multiple queues to keep track of user requests for different dynamic contents that need to be served at different target download times. Each user in the network needs to submit a request for downloading the dynamic content before the target download time.

When AoI is adopted as the performance metric, serving a user request earlier may not always improve the AoI performance of the system. This is because when a user request arrives and the specified target download time is large, it may be better to wait and obtain a more recent version than sending the current version. For example, consider the use cases when (a) an IoT device that is scheduled to perform a series of periodic tasks and requires a computational step using an NN, and (b) a car that is scheduled to start an NN-based autonomous driving application in 20 seconds. In both cases, the requests are scheduled well-ahead before the NN is required to be downloaded, *i.e.*, the target download time. Delivering the NNs early will not benefit the IoT application, since it may result in a slightly dated NN when the application actually begins. Therefore, to mitigate this, we let the users, which are the IoT applications, specify the target download time of the dynamic content. In this paper, we consider the case where the CS is scheduled to deliver the requested dynamic contents exactly at the time specified by the users, and leave the general scenario, where each user submits a time interval for dynamic content delivery, for future work. We aim to minimize the average AoI of the delivered dynamic contents. We keep track of the queues of user requests for different dynamic contents that need to be served in the near future. Since only the AoI of the dynamic contents that are delivered to the users will be accounted for system performance, we can use the information stored in the user request queues to decide when to update the cached dynamic contents in the CSs, and which dynamic content to update. As will be shown later in the paper, maintaining the user request queues and using this information can reduce the average AoI of the delivered dynamic contents in the system.

In this paper, we investigate the problem of AoI minimization of dynamic contents caching in a HetNet. Compared to previous studies on AoI in the literature, we utilize the information of the user request queues and the target download times to improve the system performance. We consider a scenario where the dynamic contents are being cached by the CSs in a HetNet. The contributions of our work are as follows:

- We formulate the problem of caching dynamic contents in a HetNet as a constrained Markov decision process (CMDP). The objective is to minimize the average AoI of the dynamic contents that are sent to the IoT applications. Compared to the existing algorithms in the literature, our proposed framework utilizes information from the queues of user requests for different dynamic contents that need to be served in the near future.
- By using the enforced decomposition technique, we pro-

pose a low-complexity suboptimal algorithm to update multiple dynamic contents being cached. In the suboptimal algorithm, a separate MDP subproblem is formulated for each cached dynamic content.
- We show that the optimal policy of the formulated CMDP for caching a single dynamic content has a threshold structure and is monotonically non-decreasing with respect to the AoI and the user request queues.
- To facilitate implementation of the proposed algorithm in practical systems, where the state space can be large for each CMDP subproblem, we train deep reinforcement learning (DRL) agents to learn the optimal policy of the formulated CMDP. This low-complexity suboptimal algorithm can effectively reduce the memory required to store the optimal policy. Our DRL is based on a state-of-the-art algorithm called deep deterministic policy gradient (DDPG) [31], which exploits the derived threshold structure of the optimal policy. The proposed DDPG algorithm achieves better convergence performance when compared to the deep Q-network (DQN) algorithm [32], which does not exploit the threshold structure.
- We perform simulations and compare the optimal policy of the CMDP problem with the offline optimal policy where all future requests are known *a priori*. We show that with the availability of information of user requests that need to be served in a few subsequent time slots, the performance of the proposed CMDP policy approaches the offline optimal policy. Furthermore, compared to the existing strategies that do not utilize the user request queues, including the periodic update approach [24], our proposed queue-aware cache content update scheduling algorithms reduce the average AoI of the dynamic contents delivered to the users by up to 30%.

The rest of this paper is organized as follows. The system model and the CMDP problem formulation are presented in Section II, where methods for obtaining the optimal solution of the CMDP problem are introduced. In Section III, we propose two low-complexity suboptimal algorithms that solve the formulated problem. Performance evaluation and comparison are presented in Section IV. Section V concludes the paper.

II. System Model and Problem Formulation

In this section, we present how the decision making module in the MBS makes the dynamic content caching decision based on the current AoI of each dynamic content and information in the user request queues. The MBS aims to minimize the long-term average AoI of all the served user requests, subject to a constraint on the update frequency of the dynamic contents. We begin this section by presenting the system model, followed by the formulation of the HetNet dynamic content caching problem as a CMDP. In particular, we introduce its state space, action space, state transition probability, constraint, and the objective function. We use a Lagrangian-based algorithm to find the optimal solution to the formulated CMDP problem, where a Lagrange multiplier that represents the cost related to updating a dynamic content is introduced. Given the Lagrange multiplier, an unconstrained MDP problem can be formulated. The corresponding deterministic stationary optimal policy can be found using relative value iteration algorithm (RVIA). A root-finding algorithm is deployed to find the optimal Lagrange multiplier, given the constraint on the maximum dynamic content update frequency. Finally, the optimal policy of the formulated CMDP is a randomized policy, which is a mixture of two deterministic stationary policies.

A. System Model

We consider a HetNet consisting of one MBS and F CSs. For the f-th CS, where $f \in \mathcal{F} = \{1, \ldots, F\}$, there are N_f users associated with it and a dynamic content is being cached. We assume only one dynamic content is being cached in each CS both for simplicity of notation and to ensure that all the CSs can operate simultaneously to serve user requests[2]. A sample system topology of the network with two CSs and two different dynamic contents being cached is shown in Fig. 1. In this example, one of the dynamic contents corresponds to the NN for navigation system for IoT-enabled cars and the other dynamic content corresponds to the NN for computer vision-based applications for reporting suspicious activities.

We consider a time-slotted system, and user requests may arrive at the beginning of each time slot. The CSs transmit the latest available cached dynamic content to the users at the beginning of the target download time via multicasting. We assume that the transmission of a dynamic content can be finished within one time slot[3], and error-free transmission can be achieved[4].

We make the simplifying assumption that newer versions of the dynamic contents become available in every time slot, which is known as the *generate-at-will* model [19] in the AoI literature. To justify this assumption, we can consider again the use case where multiple AI-enabled IoT devices perform tasks based on NNs cached in their onboard chipsets. Those NNs are trained in the cloud and need to be updated periodically and every time prior to their activation. If a stochastic gradient descent training algorithm [34] is executed in the cloud with new data added into it after each training epoch, then the IoT devices can obtain an up-to-date NN after each training step.

The CSs are connected to the MBS via a wireless backhaul. The CSs have disjoint coverage areas. Simultaneous transmissions by different CSs can be achieved when appropriate frequency reuse schemes are adopted. The channels or subcarriers used by the MBS to update the cached dynamic contents are orthogonal to those used by the CSs to serve user requests.

[2]The model can be extended to the cases where (a) multiple dynamic contents are being cached in each CS and (b) each dynamic content is being cached in multiple CSs. For case (a), spectral resources need to be allocated to each CS to ensure the user requests for different dynamic contents can be served simultaneously. For case (b), a dynamic content cached in multiple CSs can be updated via multicasting.

[3]If the dynamic contents being cached in the HetNet are NNs, where each NN is large and the transmission cannot be completed within one time slot, then the NN training algorithm, which is executed by the cloud computing server, will only update and transmit a subset of the parameters in the NN while the other parameters remain fixed. This approach is known as transfer learning [33].

[4]To consider the possibility of transmission errors, one can extend the state space by including the channel state information.

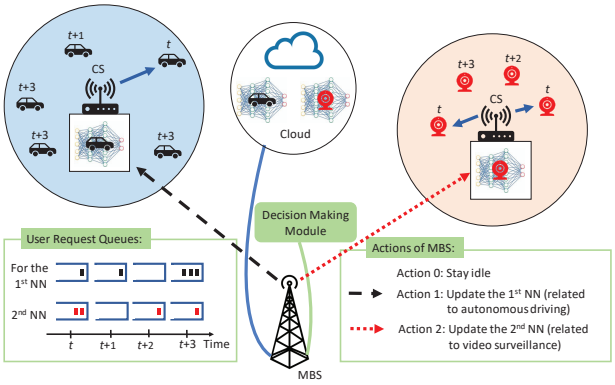


Fig. 1: System model of the HetNet with one MBS and two CSs. Two different dynamic contents are being cached, one in each CS. One of the dynamic contents corresponds to the NN for navigation system, and the other corresponds to the NN for computer vision applications. The IoT devices correspond to the IoT-enabled cars and video surveillance cameras in the network. The decision making module is located at the MBS. At a given time instance t, the target download time specified by each IoT device is shown above the IoT devices in the figure. The user request queues, which are stored in the decision making module, at time instances t, $t+1$, $t+2$, and $t+3$ are shown. The decision making module chooses one of the $(F+1)$ available actions, where action 0 corresponds to staying idle and action f corresponds to updating the f-th dynamic content, for all $f \in \mathcal{F}$.

At the beginning of each time slot, the CSs serve the user requests that are due at the end of the current time slot. The decision making module in the MBS decides whether the MBS remains idle or update one of the dynamic contents cached in a CS (see Fig. 1). We assume that apart from updating the dynamic contents cached in the CSs, the MBS also performs other tasks, such as collecting and forwarding data collected from the IoT devices. Therefore, only a certain number of time slots within a time frame can be used for dynamic content updating. We use μ, where $0 \leq \mu \leq 1$ to denote the maximum update frequency (*i.e.*, maximum fraction of time slots that are used to perform dynamic content update).

The MBS is connected to the cloud via a high-speed wired backhaul link. The IoT devices, *e.g.*, the AI-enabled cars shown in Fig. 1, submit their requests to the CS, which are forwarded to the MBS, for the latest version of NNs at least Δ time slots before the NN is required. That is, a request submitted at the t-th time slot needs to be served in the $(t+\Delta)$-th time slot. For example, Δ can be the number of time slots required by the engine and other hardware in the car to become ready for using the updated NN to perform navigation. The car submits the request when it is turned on at time slot t, and an up-to-date NN is delivered when the car is ready to be driven at time slot $t+\Delta$. Since the car will not be driven until $t+\Delta$, sending the NN too early, for example in time slot $t+1$, will lead to a larger AoI of the NN, when the car is actually ready for driving at time slot $t+\Delta$. In this paper, we consider a simple case where all users send the request Δ time slots before the dynamic contents are needed[5]. We assume that the user request arrivals of the f-th CS follow the binomial distribution with rate λ_f.

[5]In practice, the system only specifies a minimum time interval $\Delta_{\min}$, which represents the minimum time window between the time a user request is submitted and the dynamic content is needed. An IoT device may submit a request for a dynamic content which is due well ahead in the future.

B. CMDP Problem Formulation

1) Decision Epochs and States: We consider an infinite horizon CMDP, where the decision epochs are represented by the time slots in set $\mathcal{T} = \{0,1,2,\ldots\}$. In decision epoch $t \in \mathcal{T}$, let A_t^f denote the AoI of the f-th dynamic content being cached in the CS. To obtain a finite discrete state space, let $\hat{A}$ denote the upper limit of the AoI. Hence, $\mathcal{A} = \{1,2,\ldots,\hat{A}\}$ is the set of all the possible values of AoI of a cached dynamic content. In this way, we have $A_t^f \in \mathcal{A}$, $\forall\ f \in \mathcal{F}$ and $t \in \mathcal{T}$.

In decision epoch t, let $Q_t^{f,\delta} \in \mathcal{N}_f \triangleq \{0,1,\ldots,N_f\}$ denote the number of user requests for the f-th dynamic content that have their target download time at time $t+\delta$, for $f \in \mathcal{F}$ and $\delta \in \{0,\ldots,\Delta-1\}$. Since in the networking literature, *queues* are usually used to denote the number of user requests that have arrived and need to be served, we will refer to $Q_t^{f,\delta}$ as a *user request queue* in this paper. We refer to Δ as the *window size*. For example, given a window size $\Delta = 3$, in decision epoch t, the user requests that need to be served in decision epochs t, $t+1$, and $t+2$ are known. In decision epoch t, let $G_t^f \in \mathcal{N}_f$ denote the user request queue of newly arrived user requests for the f-th dynamic content, which need to be served in decision epoch $t+\Delta$.

In decision epoch t, the set of system states for the f-th dynamic content can be represented by a finite set $\mathcal{S}_f = \mathcal{A} \times \mathcal{N}_f^{\Delta} \times \mathcal{N}_f$. The state vector for the f-th dynamic content can be represented as

$$\mathbf{s}_t^f = \left(A_t^f, Q_t^{f,0}, Q_t^{f,1}, \ldots, Q_t^{f,\Delta-1}, G_t^f\right) \in \mathcal{S}_f, t \in \mathcal{T}, f \in \mathcal{F}. \tag{1}$$

As an example, in Fig. 2, we show the state of the system when $t = 6$ and $\Delta = 3$, where $\mathbf{s}_6^1 = (A_6^1, Q_6^{1,0}, Q_6^{1,1}, Q_6^{1,2}, G_6^1) = (3,1,1,0,2)$ and $\mathbf{s}_6^2 = (A_6^2, Q_6^{2,0}, Q_6^{2,1}, Q_6^{2,2}, G_6^2) = (5,2,0,1,1)$.

In summary, when considering all the F dynamic contents being cached in the HetNet, the system state space is the finite set $\mathcal{S} = \mathcal{S}_1 \times \cdots \times \mathcal{S}_F$. The state vector $\mathbf{S}_t \in \mathcal{S}$, representing the overall state of the system in decision epoch t, can be represented as

$$\begin{aligned}\mathbf{S}_t &= (\mathbf{s}_t^1, \mathbf{s}_t^2, \ldots, \mathbf{s}_t^F) \\ &= \left(A_t^1, Q_t^{1,0}, \ldots, Q_t^{1,\Delta-1}, G_t^1, \ldots, A_t^F, \right. \\ &\quad \left. Q_t^{F,0}, \ldots, Q_t^{F,\Delta-1}, G_t^F\right),\ t \in \mathcal{T}.\end{aligned} \tag{2}$$

2) Actions: Let $\mathcal{U} = \{0,1,\ldots,F\}$ denote the set of actions that can be chosen by the decision making module at the MBS. Let u_t denote the action chosen in decision epoch t, where the MBS stays idle when $u_t = 0$, updates the u_t-th dynamic content in the HetNet when $u_t > 0$.

3) State Transition Probability: Since each individual user request is independent, in decision epoch t, the number of new user request arrivals for the f-th dynamic content, G_t^f, $f \in \mathcal{F}$, follows the binomial distribution

$$\mathbb{P}(G_t^f = i) = \binom{N_f}{i} \lambda_f^i (1-\lambda_f)^{N_f - i}, \quad i \in \mathcal{N}_f. \tag{3}$$

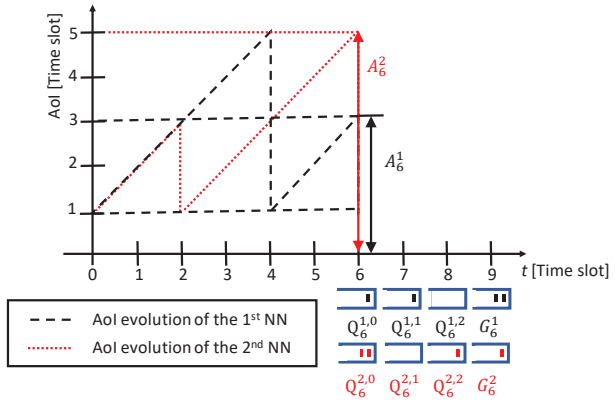


Fig. 2: Illustration of the system state in time slot $t = 6$ and window size $\Delta = 3$, where states $\mathbf{s}_6^1 = (A_6^1, Q_6^{1,0}, Q_6^{1,1}, Q_6^{1,2}, G_6^1) = (3, 1, 1, 0, 2)$ and $\mathbf{s}_6^2 = (A_6^2, Q_6^{2,0}, Q_6^{2,1}, Q_6^{2,2}, G_6^2) = (5, 2, 0, 1, 1)$.

In decision epoch $t+1$, given the AoI of the f-th dynamic content and the action chosen in decision epoch t, the AoI of the f-th dynamic content is increased by one if no update is scheduled, or reset to one if an update is scheduled for the f-th dynamic content in decision epoch t. Therefore, given the AoI A_t^f and action chosen u_t in decision epoch t, its AoI in decision epoch $t+1$ is a deterministic value, where

$$\begin{aligned}&\mathbb{P}(A_{t+1}^f \mid \mathbf{S}_t, u_t)\\ &\quad= \begin{cases} 1, & \text{if } A_{t+1}^f = A_t^f + 1 \text{ and } u_t \neq f,\\ 1, & \text{if } A_{t+1}^f = 1 \text{ and } u_t = f,\\ 0, & \text{otherwise.}\end{cases}\end{aligned} \tag{4}$$

If a user request arrives in decision epoch t, then the target download time is equal to $t + \Delta$[6]. Hence, in decision epoch $t+1$, the value of $Q_{t+1}^{f,\Delta-1}$ depends on whether a new user request arrived in decision epoch t. That is,

$$\mathbb{P}(Q_{t+1}^{f,\Delta-1} \mid \mathbf{S}_t) = \begin{cases} 1, & \text{if } Q_{t+1}^{f,\Delta-1} = G_t^f,\\ 0, & \text{otherwise.}\end{cases} \tag{5}$$

If a user request is due in $\delta+1$ time slots in decision epoch t, then this request is due in δ time slots in decision epoch $t+1$. Thus, for $Q_{t+1}^{f,\delta}$, $0 \leq \delta \leq \Delta - 2$, we have

$$\mathbb{P}(Q_{t+1}^{f,\delta} \mid \mathbf{S}_t) = \begin{cases} 1, & \text{if } Q_{t+1}^{f,\delta} = Q_t^{f,\delta+1},\\ 0, & \text{otherwise.}\end{cases} \tag{6}$$

Given the current state vector $\mathbf{S}_t$ and action u_t, the state transition probability to the next state $\mathbf{S}_{t+1}$ is equal to

$$\begin{aligned}&\mathbb{P}(\mathbf{S}_{t+1} \mid \mathbf{S}_t, u_t)\\ &= \prod_{f=1}^{F} \left(\mathbb{P}(A_{t+1}^f \mid \mathbf{S}_t, u_t) \prod_{\delta=0}^{\Delta-1} \mathbb{P}(Q_{t+1}^{f,\delta} \mid \mathbf{S}_t) \mathbb{P}(G_{t+1}^f) \right).\end{aligned} \tag{7}$$

[6]Note that a user may request a newer version of the f-th dynamic content when it already has a recent version of f. If f is not updated between the target download times of these requests, then duplicate version of f will be sent to the user.

4) Cost and Constraint: A deterministic stationary updating policy π is defined as a mapping from state space $\mathcal{S}$ to action space $\mathcal{U}$. For a system with state vector $\mathbf{S}_t$, the policy chooses an action $\pi(\mathbf{S}_t) = u_t$, $\forall\ \mathbf{S}_t \in \mathcal{S}$ and $t \in \mathcal{T}$. Similar to the approach in [27], we restrict our attention to uni-chain policies, whose induced Markov chain has a single recurrent class (and possibly some transient states) [35, vol. II, Sec. 5.2]. In [35, vol. II, Proposition 5.2.6], it is stated that for systems satisfying the weak accessibility conditions, there exists an optimal policy that is uni-chain. Since all the system states are reachable with non-zero probability, the weak accessibility conditions hold for our problem.

Let $\mathbf{S}_t^\pi$ denote the controlled Markov chain induced by policy π, where

$$\begin{aligned}\mathbf{S}_t^\pi = \big(&A_t^{1,\pi}, Q_t^{1,0,\pi}, \ldots, Q_t^{1,\Delta-1,\pi}, G_t^{1,\pi}, \ldots, A_t^{F,\pi},\\ &Q_t^{F,0,\pi}, \ldots, Q_t^{F,\Delta-1,\pi}, G_t^{F,\pi}\big),\ t \in \mathcal{T}.\end{aligned} \tag{8}$$

Note that $Q_t^{f,0,\pi}$ corresponds to the number of user requests for the f-th dynamic content that need to be served in decision epoch t. Given policy π, the sum of the expected AoI of all the served user requests in the first T decision epochs is equal to

$$M_{\text{tot}} = \sum_{t=0}^{T-1} \sum_{f=1}^{F} \mathbb{E}\left[A_t^{f,\pi} Q_t^{f,0,\pi}\right], \tag{9}$$

where $\mathbb{E}$ denotes the expectation with respect to the user request arrivals. Due to the adoption of multicasting, in decision epoch t, if there are multiple users requesting for dynamic content f, *i.e.* $Q_t^{f,0,\pi} > 1$, then these requests are being served simultaneously using one spectral resource. The average total number of user requests being served in T decision epochs is equal to

$$M_{\text{num}} = T \sum_{f=1}^{F} N_f \lambda_f. \tag{10}$$

Hence, given policy π, the average AoI and the update frequency of all user requests are as follows:

$$\overline{M}(\pi) = \limsup_{T\to\infty} \frac{M_{\text{tot}}}{M_{\text{num}}} \tag{11a}$$

$$= \limsup_{T\to\infty} \frac{\sum_{t=0}^{T-1} \sum_{f=1}^{F} \mathbb{E}\left[A_t^{f,\pi} Q_t^{f,0,\pi}\right]}{T \sum_{f=1}^{F} N_f \lambda_f}, \tag{11b}$$

$$\overline{C}(\pi) = \limsup_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}\left[\mathbf{I}(\pi(\mathbf{S}_t^\pi) > 0)\right], \tag{12}$$

where $\mathbf{S}_t^\pi$, $\forall\ t \in \mathcal{T}$, follows the state transition probability specified in (7), and $\mathbf{I}(\cdot)$ is the indicator function.

The optimal policy π^* is defined to be a uni-chain policy that minimizes the average cost, which corresponds to the average AoI of the served user requests, while satisfying a constraint on the update frequency of the dynamic content. That is,

$$\begin{aligned}\overline{M}^* = \underset{\pi}{\text{minimize}}\ \ &\overline{M}(\pi)\\ \text{subject to}\ \ &\overline{C}(\pi) \leq \mu.\end{aligned} \tag{13}$$

Problem (13) is an infinite horizon average cost CMDP problem [36], [37]. In the following subsection, we use a Lagrangian-based method [27], [30] to find the optimal solution to the CMDP problem.

C. Optimal Solution of the CMDP

Following the Lagrangian approach to solve the formulated average cost CMDP problem (13), we define the Lagrangian for a policy π, given a Lagrange multiplier $\eta \geq 0$, as

$$\overline{L}(\eta,\pi) \triangleq \limsup_{T\to\infty}\frac{1}{T}\Bigg(\sum_{t=0}^{T-1}\frac{\sum_{f=1}^{F}\mathbb{E}\left[A_t^{f,\pi}Q_t^{f,0,\pi}\right]}{\sum_{f=1}^{F}N_f\lambda_f} + \eta\sum_{t=0}^{T-1}\mathbb{E}\left[\mathbf{I}(\pi(\mathbf{S}_t^{\pi})>0)\right]\Bigg). \tag{14}$$

We define the Lagrangian in decision epoch t as

$$c(\mathbf{S}_t,u_t,\eta) \triangleq \frac{\sum_{f=1}^{F}A_t^f Q_t^{f,0}}{\sum_{f=1}^{F}N_f\lambda_f} + \eta\mathbf{I}(u_t>0). \tag{15}$$

The Lagrangian in (14) can be re-written as

$$\overline{L}(\eta,\pi) = \limsup_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\left[c(\mathbf{S}_t^{\pi},\pi(\mathbf{S}_t^{\pi}),\eta)\right]. \tag{16}$$

Given a Lagrange multiplier η, we can determine an optimal policy that minimizes the Lagrangian. Let $\overline{L}^*(\eta)$ denote the optimal Lagrangian for Lagrange multiplier $\eta \geq 0$, where

$$\overline{L}^*(\eta) = \min_{\pi}\ \overline{L}(\eta,\pi), \tag{17}$$

and

$$\pi_\eta^* = \arg\min_{\pi}\ \overline{L}(\eta,\pi). \tag{18}$$

We refer to the policy that satisfies (18) as the η-optimal policy. To obtain the η-optimal policy π_η^* for a given Lagrange multiplier η in an unconstrained MDP problem, we use the differential Bellman optimality equations introduced in [27], [35, vol. II, Propositions 5.2.1 and 5.2.2].

Lemma 1. *For any $\eta \geq 0$, there exists $(\theta_\eta, J(\mathbf{S}_t,\eta))$ satisfying*

$$\theta_\eta + J(\mathbf{S}_t,\eta) = \min_{u_t\in\mathcal{U}}\Bigg\{c(\mathbf{S}_t,u_t,\eta) + \sum_{\mathbf{S}_{t+1}\in\mathcal{S}}\mathbb{P}(\mathbf{S}_{t+1}\mid\mathbf{S}_t,u_t)J(\mathbf{S}_{t+1},\eta)\Bigg\}, \tag{19}$$

where $\mathbf{S}_{t+1}$ satisfies the state transition probability (7), and $\theta_\eta = \overline{L}^*(\eta)$ for all initial states. $J(\mathbf{S}_t,\eta)$ is the cost-to-go function, and represents the expected total relative future cost starting from a given state $\mathbf{S}_t \in \mathcal{S}$. The η-optimal policy achieving the optimal Lagrangian $\overline{L}^*(\eta)$ can be found as

$$\pi_\eta^*(\mathbf{S}_t) = \arg\min_{u_t\in\mathcal{U}}\Bigg\{c(\mathbf{S}_t,u_t,\eta) + \sum_{\mathbf{S}_{t+1}\in\mathcal{S}}\mathbb{P}(\mathbf{S}_{t+1}\mid\mathbf{S}_t,u_t)J(\mathbf{S}_{t+1},\eta)\Bigg\}. \tag{20}$$

The proof of Lemma 1 can be found in [35, vol. II, Proposition 5.1.4].

Algorithm 1 Relative Value Iteration Algorithm (RVIA)

Input: η, $\boldsymbol{\lambda} = [\lambda_1, \lambda_2, \ldots, \lambda_F]$, $\boldsymbol{N} = [N_1, \ldots, N_F]$, Δ, η, and ϵ_r

Arbitrarily choose a reference state $\mathbf{S}^{\text{ref}} \in \mathcal{S}$
$J_0(\mathbf{S},\eta) \leftarrow 0, \quad \forall\ \mathbf{S}\in\mathcal{S}$
for $\mathbf{S}\in\mathcal{S}$ **do**
  for $u\in\mathcal{U}$ **do**
    $V_1(\mathbf{S},u,\eta) \leftarrow c(\mathbf{S},u,\eta) + \sum_{\mathbf{S}_t\in\mathcal{S}}\mathbb{P}(\mathbf{S}_t\mid\mathbf{S},u)J_0(\mathbf{S}_t,\eta)$
  end for
  $H_1(\mathbf{S},\eta) \leftarrow \min_{u\in\mathcal{U}} V_1(\mathbf{S},u,\eta)$
end for
$J_1(\mathbf{S},\eta) \leftarrow H_1(\mathbf{S},\eta) - H_1(\mathbf{S}^{\text{ref}},\eta),\ \forall\ \mathbf{S}\in\mathcal{S}$
$k \leftarrow 1$
while $\exists\ \mathbf{S}\in\mathcal{S}$, such that $|J_k(\mathbf{S},\eta) - J_{k-1}(\mathbf{S},\eta)| > \epsilon_r$ **do**
  for $\mathbf{S}\in\mathcal{S}$ **do**
    for $u\in\mathcal{U}$ **do**
      $V_{k+1}(\mathbf{S},u,\eta) \leftarrow c(\mathbf{S},u,\eta) + \sum_{\mathbf{S}_k\in\mathcal{S}}\mathbb{P}(\mathbf{S}_k\mid\mathbf{S},u)J_k(\mathbf{S}_k,\eta)$
    end for
    $H_{k+1}(\mathbf{S},\eta) \leftarrow \min_{u\in\mathcal{U}} V_{k+1}(\mathbf{S},u,\eta)$
  end for
  $J_{k+1}(\mathbf{S},\eta) \leftarrow H_{k+1}(\mathbf{S},\eta) - H_{k+1}(\mathbf{S}^{\text{ref}},\eta),\ \forall\ \mathbf{S}\in\mathcal{S}$
  $k \leftarrow k+1$
end while
$\pi_\eta^*(\mathbf{S}) \leftarrow \arg\min_{u\in\mathcal{U}} V_k(\mathbf{S},u,\eta),\ \forall\ \mathbf{S}\in\mathcal{S}$
$V(\mathbf{S},u,\eta) \leftarrow V_k(\mathbf{S},u,\eta),\ \forall\ \mathbf{S}\in\mathcal{S},\ u\in\mathcal{U}$
$J(\mathbf{S},\eta) \leftarrow J_k(\mathbf{S},\eta),\ \forall\ \mathbf{S}\in\mathcal{S}$
return Optimal policy $\pi_\eta^*(\mathbf{S})$, $V(\mathbf{S},u,\eta),\ \forall\ \mathbf{S}\in\mathcal{S},\ u\in\mathcal{U}$ and $J(\mathbf{S},\eta),\ \forall\ \mathbf{S}\in\mathcal{S}$

For a given η, we define the state-action value function $V(\mathbf{S}_t,u_t,\eta)$ as follows:

$$V(\mathbf{S}_t,u_t,\eta) \triangleq c(\mathbf{S}_t,u_t,\eta) + \sum_{\mathbf{S}_{t+1}\in\mathcal{S}}\mathbb{P}(\mathbf{S}_{t+1}\mid\mathbf{S}_t,u_t) \times J(\mathbf{S}_{t+1},\eta), \quad \mathbf{S}_t\in\mathcal{S}, u_t\in\mathcal{U}. \tag{21}$$

In practice, finding $J(\mathbf{S}_t,\eta)$ and $V(\mathbf{S}_t,u_t,\eta)$ involve solving the Bellman equations iteratively, using methods such as RVIA [35] (shown in Algorithm 1). From the RVIA, the obtained optimal policy needs to be saved in the memory of the decision making module. The module chooses the optimal action based on the value stored in a look-up table. However, due to the *curse of dimensionality*, the size of the stored optimal policy matrix grows exponentially as the state space expands.

From [36, Lemma 3.1], $\overline{C}(\pi_\eta^*)$ in (12) is a non-decreasing function with respect to η. Therefore, to find the optimal policy π^* for problem (13), we can determine the Lagrange multiplier η^*, such that

$$\eta^* = \inf\{\eta\geq 0 \mid \overline{C}(\pi_\eta^*) \leq \mu\}. \tag{22}$$

In practice, finding η^* requires iterative numerical methods, such as the Robbins-Monro algorithm [38]. After η^* has

been determined, the optimal solution to the CMDP (13) is a randomized mixture of two deterministic policies, as stated in Lemma 2 [37] below.

Lemma 2. *The optimal policy π^* of the formulated CMDP (13) is a randomized mixture of two deterministic stationary policies $\pi^*_{\eta^*,1}$ and $\pi^*_{\eta^*,2}$, in the form*

$$\pi^* = \alpha\pi^*_{\eta^*,1} + (1-\alpha)\pi^*_{\eta^*,2}, \tag{23}$$

where $\alpha \in [0,1]$ is a randomization parameter. $\pi^*_{\eta^*,1}$ and $\pi^*_{\eta^*,2}$ are the optimal policies of the unconstrained MDP under the Lagrange multiplier η^*, where $\pi^*_{\eta^*,1}$ and $\pi^*_{\eta^*,2}$ differ at most in a single state $\mathbf{S}_t \in \mathcal{S}$.

The proof of Lemma 2 can be found in [37]. Although Lemma 2 suggests the existence of the optimal policy of the CMDP, finding this policy is not computationally tractable. The first aspect is that the state space of the CMDP problem can be large, especially when a large number of dynamic contents are being cached. This means that the dynamic programming-based RVIA algorithm, which is only suitable for solving problems with a small state space, quickly becomes computationally intractable. The second aspect is that finding η^*, $\pi^*_{\eta^*,1}$, and $\pi^*_{\eta^*,2}$ is computationally demanding. It has been reported that the convergence speed of the iterative algorithm for finding η can be slow [27], [30]. In the next section, we will investigate suboptimal solutions that have lower computational complexity.

III. Suboptimal Algorithms Design

In this section, we develop two suboptimal algorithms with lower computational complexity. To address the large state space of the CMDP problem (13), we decompose the CMDP problem into F subproblems, based on the principle of enforced decomposition [35]. When the state space of each subproblem becomes large, we address this by using DRL.

A. Enforced Decomposition

1) Decomposition into Subproblems: Consider a special case of the system where only one particular dynamic content $f \in \mathcal{F}$ is being cached in the HetNet. The state space of the system is $\mathcal{S}_f$. In this case, the MBS only has two actions from its action set $\mathcal{U}_F = \{0,1\}$ to choose from: action $u_t^f = 0$ corresponds to staying idle, and action $u_t^f = 1$ corresponds to updating the f-th dynamic content.

Given $\mathbf{s}_t^f \in \mathcal{S}_f$ and $u_t^f \in \mathcal{U}_F$, the state transition probability of the system can be written as

$$\begin{aligned}\mathbb{P}(\mathbf{s}_{t+1}^f \mid \mathbf{s}_t^f, u_t^f) =& \mathbb{P}(A_{t+1}^f \mid \mathbf{s}_t^f, u_t^f) \\ &\times \prod_{\delta=0}^{\Delta-1} \mathbb{P}(Q_{t+1}^{f,\delta} \mid \mathbf{s}_t^f)\mathbb{P}(G_{t+1}^f),\end{aligned} \tag{24}$$

where $\mathbb{P}(G_{t+1}^f)$, $\mathbb{P}(Q_{t+1}^{f,\delta} \mid \mathbf{s}_t^f)$, $\delta \in \{0,1,\dots\Delta-1\}$ are defined in Section II-B. We have

$$\mathbb{P}(A_{t+1}^f \mid \mathbf{s}_t^f, u_t^f) = \begin{cases} 1, & \text{if } A_{t+1}^f = A_t^f + 1 \text{ and } u_t^f = 0, \\ 1, & \text{if } A_{t+1}^f = 1 \text{ and } u_t^f = 1, \\ 0, & \text{otherwise.} \end{cases} \tag{25}$$

We define a stationary updating policy π^f as a mapping from $\mathcal{S}_f$ to $\mathcal{U}_F$. For a system with state vector $\mathbf{s}_t^f$, the policy chooses action $\pi^f(\mathbf{s}_t^f) = u_t^f$, for all $\mathbf{s}_t^f \in \mathcal{S}_f$ and $t \in \mathcal{T}$. Given policy π^f, the average AoI and update frequency of the system when only the f-th dynamic content is being cached can be found as

$$\overline{M}^f(\pi^f) = \limsup_{T\to\infty} \frac{1}{TN_f\lambda_f} \sum_{t=0}^{T-1} \mathbb{E}\left[A_t^{f,\pi^f} Q_t^{f,0,\pi^f}\right], \tag{26}$$

$$\overline{C}^f(\pi^f) = \limsup_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}\left[\mathbf{I}\left(\pi(\mathbf{s}_t^{f,\pi^f}) = 1\right)\right], \tag{27}$$

where $\mathbf{s}_t^{f,\pi^f}$, for all $t \in \mathcal{T}$, follows the state transition probability specified in (24).

Given μ, N_f and λ_f, for all $f \in \mathcal{F}$, Yates *et al.* in [24] derived the following optimal update frequency for dynamic content f when the scheduling decision is made without prior knowledge of the user request queues (*i.e.*, $\Delta = 0$ in our problem),

$$\mu_f = \frac{\mu\sqrt{N_f\lambda_f}}{\sum_{i=1}^{F}\sqrt{N_i\lambda_i}}, \quad f \in \mathcal{F}. \tag{28}$$

In finding a low-complexity suboptimal policy of the enforced decomposition solution with good performance, we adopt the above update frequency for the cases $\Delta > 0$. In this way, the optimal policy $\pi^{f,*}$ can be determined by solving the following CMDP problem

$$\begin{aligned}\overline{M}^{f,*} = \underset{\pi^f}{\text{minimize}} \;\; & \overline{M}^f(\pi^f) \\ \text{subject to} \;\; & \overline{C}^f(\pi^f) \le \mu_f.\end{aligned} \tag{29}$$

Given a Lagrange multiplier $\eta_f \ge 0$, the Lagrangian in each decision epoch t can be found as

$$c^f(\mathbf{s}_t^f, u_t^f, \eta_f) = \frac{A_t^{f,\pi^f} Q_t^{f,0,\pi^f}}{N_f\lambda_f} + \eta_f \mathbf{I}(u_t^f = 1). \tag{30}$$

The expectation of the Lagrangian using policy π^f becomes

$$\overline{L}(\eta_f, \pi^f) = \limsup_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}\left[c^f\left(\mathbf{s}_t^{f,\pi^f}, \pi^f(\mathbf{s}_t^{f,\pi^f}), \eta_f\right)\right]. \tag{31}$$

Using the same RVIA algorithm shown in Algorithm 1, and consider the original CMDP problem where only the f-th dynamic content is being cached (by setting $\mathbf{S}_t = \mathbf{s}_t^f$), we can determine $\theta_{\eta_f}^f$, $J^f(\mathbf{s}_t^f, \eta_f)$, $V^f(\mathbf{s}_t^f, u_t^f, \eta_f)$, and $\pi_{\eta_f}^{f,*}$ satisfying

$$\begin{aligned}\theta_{\eta_f}^f + J^f(\mathbf{s}_t^f, \eta_f) = \min_{u_t^f \in \mathcal{U}_F} \Bigg\{ & c^f(\mathbf{s}_t^f, u_t^f, \eta_f) \\ & + \sum_{\mathbf{s}_{t+1}^f \in \mathcal{S}_f} \mathbb{P}(\mathbf{s}_{t+1}^f \mid \mathbf{s}_t^f, u_t^f) J^f(\mathbf{s}_{t+1}^f, \eta_f) \Bigg\},\end{aligned} \tag{32}$$

$$\begin{aligned}V^f(\mathbf{s}_t^f, u_t^f, \eta_f) = c^f(\mathbf{s}_t^f, u_t^f, \eta_f) + \sum_{\mathbf{s}_{t+1}^f \in \mathcal{S}_f} & \mathbb{P}(\mathbf{s}_{t+1}^f \mid \mathbf{s}_t^f, u_t^f) \\ & \times J^f(\mathbf{s}_{t+1}^f, \eta_f),\end{aligned} \tag{33}$$

Algorithm 2 Enforced Decomposition Algorithm

Input: $\eta_1^*, \ldots, \eta_F^*$, $\lambda_1, \lambda_2, \ldots, \lambda_F$, $N_1, \ldots, N_F$, Δ, and ϵ_r
for $f \in \mathcal{F}$ **do**
  Set $\boldsymbol{\lambda} \leftarrow \lambda_f$ and $\boldsymbol{N} \leftarrow N_f$
  Call Algorithm 1 to determine $V(\mathbf{s}^f, u^f, \eta_f^*)$
  for $u^f \in \mathcal{U}_F$ **do**
    $V^f(\mathbf{s}^f, u^f, \eta_f^*) \leftarrow V(\mathbf{s}^f, u^f, \eta_f^*), \quad \mathbf{s}^f \in \mathcal{S}_f$
  end for
end for
$\tilde{\pi}^*(\mathbf{S}) \leftarrow \arg\min_{u \in \mathcal{U}} \sum_{f \in \mathcal{F}} V^f(\mathbf{s}^f, \mathbf{I}(u = f), \eta_f^*), \quad \mathbf{S} \in \mathcal{S}$
return Optimal policy $\tilde{\pi}^*$

and

$$\pi_{\eta_f}^{f,*}(\mathbf{s}_t^f) = \arg\min_{u_t^f \in \mathcal{U}_F} V^f(\mathbf{s}_t^f, u_t^f, \eta_f). \tag{34}$$

The optimal Lagrange multiplier η_f^* satisfies

$$\eta_f^* = \inf\{\eta_f \geq 0 \mid \overline{C}^f(\pi_{\eta_f}^*) \leq \mu_f\}. \tag{35}$$

Now, consider the case where all F dynamic contents are being cached in the system. After obtaining $V^f(\mathbf{s}_t^f, u_t^f, \eta_f^*)$, $\forall f \in \mathcal{F}$, the action is chosen by

$$\tilde{\pi}^*(\mathbf{S}_t) = \arg\min_{u_t \in \mathcal{U}} \sum_{f \in \mathcal{F}} V^f(\mathbf{s}_t^f, \mathbf{I}(u_t = f), \eta_f^*). \tag{36}$$

The decomposition step in (36) follows the enforced decomposition technique introduced in [35]. By using the decomposed optimal policies $\pi_{\eta_f^*}^{f,*}$ to approximate the optimal policy with complete state space, we train F individual CMDP processes, one for each of the F dynamic contents. During the training of each dynamic content, we consider a HetNet where the particular dynamic content under consideration is requested by the users. For computational tractability, we propose to find a feasible Lagrange multiplier for each dynamic content that satisfies the constraint in problem (29). This can be achieved using classical root-finding algorithms, such as bisection search or the Robbins-Monro algorithm [38]. Using these approximation steps, the state space of the CMDP problem grows linearly, instead of exponentially, with respect to F (*i.e.*, the number of files being cached). The gain is two-fold: First, the memory space required for storing the look-up table for the state-action value function is reduced. Secondly, the algorithmic complexity of the RVIA algorithm is also reduced.

B. Structural Property of $\pi_{\eta_f}^{f,*}$

In this subsection, we investigate the monotonicity and submodular properties of function $V^f(\mathbf{s}_t^f, u_t^f, \eta_f)$. Based on this, we prove that the optimal policy in problem (29) has a threshold structure, which will be used to design a policy-gradient-based DRL algorithm in the next subsection.

By using the RVIA, the state transition probability (24), and the Lagrangian (30), we can prove the following properties.

Lemma 3. *(Monotonicity of $J^f(\mathbf{s}_t^f, \eta_f)$ and $V^f(\mathbf{s}_t^f, u_t^f, \eta_f)$): Given $\eta_f \geq 0$, $J^f(\mathbf{s}_t^f, \eta_f)$ and $V^f(\mathbf{s}_t^f, u_t^f, \eta_f)$ are monotonically non-decreasing functions with respect to A_t^f, G_t^f, and $Q_t^{f,\delta}$, $\forall \delta \in \{0, \ldots, \Delta - 1\}$.*

Proof: See Appendix A. ∎

In decision epoch t with state vector $\mathbf{s}_t^f$, for any two actions $u_1^f, u_2^f \in \mathcal{U}_F$, let us define an auxiliary function $\Gamma_{u_1^f, u_2^f}^f : \mathcal{S}_f \times \mathbb{R}^+ \mapsto \mathbb{R}$

$$\Gamma_{u_1^f, u_2^f}^f(\mathbf{s}_t^f, \eta_f) \triangleq V^f(\mathbf{s}_t^f, u_1^f, \eta_f) - V^f(\mathbf{s}_t^f, u_2^f, \eta_f). \tag{37}$$

An action u_1^f is said to dominate action u_2^f in state $\mathbf{s}_t^f$ if $\Gamma_{u_1^f, u_2^f}^f(\mathbf{s}_t^f, \eta_f) \leq 0$, and action u_1^f is referred to as the dominating action in state $\mathbf{s}_t^f$. We can prove the following monotonicity result.

Lemma 4. *(Monotonicity of $\Gamma_{1,0}^f(\mathbf{s}_t^f, \eta_f)$) For any state $\mathbf{s}_t^f \in \mathcal{S}_f$, $\Gamma_{1,0}^f(\mathbf{s}_t^f, \eta_f)$ is a monotonically non-increasing function in A_t^f.*

Proof: See Appendix B. ∎

In the literature, Lemma 4 is also known as the submodular property of $V^f(\mathbf{s}_t^f, u_t^f, \eta_f)$, which implies that $V^f(\mathbf{s}_t^f, u_t^f, \eta_f)$ is submodular in state $\mathbf{s}_t^f$ and action u_t^f. Let us define the set

$$\begin{aligned} &\Phi^f(Q_t^{f,0}, \ldots, Q_t^{f,\Delta-1}, G_t^f, \eta_f) \\ &\qquad \triangleq \{A_t^f \mid A_t^f \in \mathcal{A},\ \Gamma_{1,0}^f(\mathbf{s}_t^f, \eta_f) \leq 0\}. \end{aligned} \tag{38}$$

We also define

$$\begin{aligned} &\phi^f(Q_t^{f,0}, \ldots, Q_t^{f,\Delta-1}, G_t^f, \eta_f) \\ &\triangleq \begin{cases} \min\left[\Phi^f(Q_t^{f,0}, \ldots, Q_t^{f,\Delta-1}, G_t^f, \eta_f)\right], \\ \qquad \text{if } \Phi^f(Q_t^{f,0}, \ldots, Q_t^{f,\Delta-1}, G_t^f, \eta_f) \neq \emptyset \\ -\infty, \quad \text{otherwise}, \end{cases} \end{aligned} \tag{39}$$

where $\min[\cdot]$ denotes the minimum element in a set.

Theorem 1. *(Structural property of $\pi_{\eta_f}^{f,*}$) Given $\eta_f \geq 0$, for any state $\mathbf{s}_t^f \in \mathcal{S}_f$, the optimal policy $\pi_{\eta_f}^{f,*}$ has the following threshold structure*

$$\pi_{\eta_f}^{f,*}(\mathbf{s}_t^f) = \begin{cases} 1, & \text{if } A_t^f \geq \phi^f(Q_t^{f,0}, \ldots, Q_t^{f,\Delta-1}, G_t^f, \eta_f) \\ 0, & \text{otherwise}. \end{cases} \tag{40}$$

Proof: See Appendix C. ∎

To simplify the notation, we define $\tilde{\mathbf{s}}_t^f = (Q_t^{f,0}, \ldots, Q_t^{f,\Delta-1}, G_t^f)$, which specifies the system state in decision epoch $t \in \mathcal{T}$, excluding A_t^f. Theorem 1 shows that given $\tilde{\mathbf{s}}_t^f$, the optimal action can be chosen based on whether $A_t^f \geq \phi^f(\tilde{\mathbf{s}}_t^f, \eta_f)$. In the next subsection, we will exploit this threshold structure and train an NN to estimate $\phi^f(\tilde{\mathbf{s}}_t^f, \eta_f)$. Compared to learning $V^f(\mathbf{s}_t^f, u_t^f, \eta_f)$, the input dimension of the NN is reduced and the training process converges faster.

C. Deep Reinforcement Learning (DRL)

Despite the reduction of complexity brought by enforced decomposition, finding the optimal values of the state-action value function still requires using the RVIA, and the optimal state-action values $V^f(\mathbf{s}_t^f, u_t^f, \eta_f)$, $\forall\ \mathbf{s}_t^f \in \mathbf{S}_f$, $u_t^f \in \mathcal{U}_F$, $f \in \mathcal{F}$ need to be stored in the memory. The memory required may still be large for implementation in practical systems, as the potential number of users for each dynamic content, N_f, $f \in \mathcal{F}$, and $\hat{A}$ become large.

Leveraging the recent development in AI, in this subsection, we propose to estimate the threshold of updating a dynamic content, $\phi^f(\tilde{\mathbf{s}}_t^f, \eta_f)$, using a policy-gradient DRL-based method. The policy is approximated by an NN, called the *actor network*, which takes the state of the CMDP as input, and its output corresponds to the threshold value. Since the optimal policy has a threshold structure as specified in (40), the action can be chosen based on the output of the actor network. Let $\phi_t^f(\cdot \mid \boldsymbol{\xi}_t)$ denote the threshold function approximated by the actor network with parameters $\boldsymbol{\xi}_t$. At training time step t, given $\tilde{\mathbf{s}}_t^f$ and η_f, the threshold is chosen as

$$\psi_t^f = \phi_t^f(\tilde{\mathbf{s}}_t^f, \eta_f \mid \boldsymbol{\xi}_t) + \text{Norm}(\sigma^2), \tag{41}$$

where $\text{Norm}(\sigma^2)$ denotes the independent and identically distributed Gaussian noise with zero mean and variance σ^2. Thereafter, based on (40), the action taken is $u_t^f = \mathbf{I}(A_t^f \geq \psi_t^f)$.

We adopt a state-of-the-art DRL algorithm called DDPG [31] to approximate the threshold function, where another NN called the *critic network* is jointly trained to reduce the variance of the training process. At training time step t, when the system state is $\mathbf{s}_t^f$, let $\tilde{V}_t^f(\mathbf{s}_t^f, \psi_t^f, \eta_f)$ denote the state-action value function corresponding to choosing ψ_t^f as the threshold for updating the dynamic content. Let $\tilde{V}_t^f(\cdot \mid \boldsymbol{\theta}_t)$ denote the state-action value function approximated by the critic network with parameters $\boldsymbol{\theta}_t$ at training time step t. Accounting for the complexity, training time, and the accuracy in approximation, we design an actor network with two hidden layers with sizes 512 and 128. The rectified linear units (ReLUs) are used as the activation functions for the hidden layers, while a $\tanh(\cdot)$ activation function and scaling are used at the output layer, to ensure that the threshold ψ_t^f is within the range $[1, \hat{A}]$. We design a critic network with three hidden layers, with sizes 1024, 512, and 300 for each layer. Similarly, ReLUs are used as the activation functions for the hidden layers of the critic network.

We divide the training process into N_{epi} episodes to track the training performance, where each episode contains T_{epi} *training time steps*. Therefore, there are in total $N_{\text{epi}}T_{\text{epi}}$ training time steps, represented by the set $\mathcal{T}_{\text{train}} = \{1, \ldots, N_{\text{epi}}T_{\text{epi}}\}$. To improve the stability of the training process, an NN with the same dimensions as the actor network, called the *target actor network*, is created and being updated during the training steps. Similarly, an NN with the same dimensions as the critic network, called the *target critic network*, is created and being updated. Let $\phi_t^f(\cdot \mid \boldsymbol{\xi}_t^{\text{target}})$ denote the target function approximated by the target actor network with parameters $\boldsymbol{\xi}_t^{\text{target}}$ and let $\tilde{V}_t^f(\cdot \mid \boldsymbol{\theta}_t^{\text{target}})$ denote the state-action value function approximated by the target critic network with parameters $\boldsymbol{\theta}_t^{\text{target}}$.

In Monte Carlo reinforcement learning, sampled experience (*i.e.*, simulated interaction with an environment) is used to estimate the state-action value functions. Given current state $\mathbf{s}_t^f$ and decision epoch t, we use a simulator to sample the next system state $\mathbf{s}_{t+1}^f$ in decision epoch $t+1$ and the Lagrangian $c^f(\mathbf{s}_t^f, \mathbf{I}(A_t^f \geq \psi_t^f), \eta_f)$. The state-action value function update step for an average-cost MDP problem is

$$\begin{aligned}\tilde{V}_{t+1}^f(\mathbf{s}_t^f, \psi_t^f, \eta_f) \ =\ & \tilde{V}_t^f\left(\mathbf{s}_t^f, \psi_t^f, \eta_f\right) \\ & + \beta\Big(c^f(\mathbf{s}_t^f, \mathbf{I}(A_t^f \geq \psi_t^f), \eta_f) \\ & + \min_{\psi^* \in [1, \hat{A}]} \tilde{V}_t^f(\mathbf{s}_{t+1}^f, \psi^*, \eta_f) \\ & - \min_{\psi^* \in [1, \hat{A}]} \tilde{V}_t^f(\mathbf{s}^{f,\text{ref}}, \psi^*, \eta_f) - \tilde{V}_t^f(\mathbf{s}_t^f, \psi_t^f, \eta_f)\Big),\end{aligned} \tag{42}$$

where β is the learning rate parameter and $\mathbf{s}^{f,\text{ref}}$ is a fixed state that can be chosen arbitrarily and remains fixed during the entire training process [35]. Given $\boldsymbol{\theta}_t$, the loss is defined as

$$\begin{aligned}& L(\boldsymbol{\theta}_t) \\ &= \frac{1}{2}\Big(\tilde{V}_t^f\left(\mathbf{s}_{t+1}^f, \phi^f(\tilde{\mathbf{s}}_{t+1}^f, \eta_f \mid \boldsymbol{\xi}_t^{\text{target}}), \eta_f \mid \boldsymbol{\theta}_t^{\text{target}}\right) \\ &\quad - \tilde{V}_t^f\left(\mathbf{s}^{f,\text{ref}}, \phi^f(\tilde{\mathbf{s}}_t^{f,\text{ref}}, \eta_f \mid \boldsymbol{\xi}_t^{\text{target}}), \eta_f \mid \boldsymbol{\theta}_t^{\text{target}}\right) \\ &\quad - \tilde{V}_t^f\left(\mathbf{s}_t^f, \phi^f(\tilde{\mathbf{s}}_t^f, \eta_f \mid \boldsymbol{\xi}_t^{\text{target}}), \eta_f \mid \boldsymbol{\theta}_t\right) \\ &\quad + c^f(\mathbf{s}_t^f, \mathbf{I}(A_t^f \geq \psi_t^f), \eta_f)\Big)^2.\end{aligned} \tag{43}$$

The actor policy network is updated based on the deterministic policy gradient theorem [31]

$$\nabla_{\boldsymbol{\xi}_t} J(\boldsymbol{\xi}_t) = \mathbb{E}\left[\nabla_{\psi_t^f} \tilde{V}_t^f(\mathbf{s}_t^f, \psi_t^f, \eta_f \mid \boldsymbol{\theta}_t) \nabla_{\boldsymbol{\xi}_t} \phi^f(\tilde{\mathbf{s}}_t^f, \eta_f \mid \boldsymbol{\xi}_t)\right]. \tag{44}$$

To reduce the degree of correlation among the observed sequence of data and to improve the stability of DRL, we adopt the *experience replay* approach, where the *system transition tuples* $(\mathbf{s}_t^f, \psi_t^f, c^f(\mathbf{s}_t^f, \mathbf{I}(A_t^f \geq \psi_t^f), \eta_f), \mathbf{s}_{t+1}^f)$ are stored in the *replay memory* after each training time step t. At training time step t, a set of K_{batch} system transition tuples $\mathcal{K}_t^{\text{batch}}$ are randomly drawn from the replay memory, and *batch gradient descent* [32] is employed to minimize the sum of the loss functions of all the K_{batch} system transition tuples. A stochastic gradient descent step can be expressed as

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \beta_s \nabla_{\boldsymbol{\theta}_t} L(\boldsymbol{\theta}_t), \tag{45}$$

$$\boldsymbol{\xi}_{t+1} = \boldsymbol{\xi}_t - \beta_s \nabla_{\boldsymbol{\xi}_t} J(\boldsymbol{\xi}_t), \tag{46}$$

where β_s is the learning rate for stochastic gradient descent. A soft update approach is used to update parameters of the target actor network and the target critic network. To obtain the suboptimal policy (36) for $F > 1$, we use the approximation

$$V_t^f(\mathbf{s}_t^f, 0, \eta_f) \approx \tilde{V}_t^f(\mathbf{s}_t^f, \min(\hat{A}, \mathbf{s}_t^f + \rho), \eta_f \mid \boldsymbol{\theta}_t), \tag{47}$$

$$V_t^f(\mathbf{s}_t^f, 1, \eta_f) \approx \tilde{V}_t^f(\mathbf{s}_t^f, \max(1, \mathbf{s}_t^f - \rho), \eta_f \mid \boldsymbol{\theta}_t), \tag{48}$$

Algorithm 3 Deep Deterministic Policy Gradient (DDPG) Algorithm

Input: η_f, τ, β_s, σ^2, N_{epi}, T_{epi}, T_{update}, and K_{batch}
Initialize the replay memory
Initialize the actor network parameters $\boldsymbol{\theta}_0$ and target actor network parameters $\boldsymbol{\theta}_0^{\text{target}}$
Initialize critic network parameters $\boldsymbol{\xi}_0$, target critic network parameters $\boldsymbol{\xi}_0^{\text{target}}$
Observe the initial state $\mathbf{s}_0^f$ and select a random threshold ψ_0^f
for $t \in \mathcal{T}_{\text{train}}$ **do**
    System samples $\mathbf{s}_t^f$ and $c^f(\mathbf{s}_t^f, \mathbf{I}(A_t^f \geq \psi_t^f), \eta_f)$
    Save the system transition tuple to the replay memory
    Choose threshold according to (41)
    Randomly sample a set of $\mathcal{K}_t^{\text{batch}}$ system transition tuples from the replay memory
    for $j \in \mathcal{K}_t^{\text{batch}}$ **do**
        Calculate the loss function based on (43)
        Calculate the policy gradient based on (44)
    end for
    Update critic network parameter $\boldsymbol{\theta}_t$ based on (45)
    Update actor network parameter $\boldsymbol{\xi}_t$ based on (46)
    $\boldsymbol{\theta}_t^{\text{target}} \leftarrow \tau\boldsymbol{\theta}_t + (1-\tau)\boldsymbol{\theta}_t^{\text{target}}$
    $\boldsymbol{\xi}_t^{\text{target}} \leftarrow \tau\boldsymbol{\xi}_t + (1-\tau)\boldsymbol{\xi}_t^{\text{target}}$
end for
$\boldsymbol{\theta}^{\text{target}} \leftarrow \boldsymbol{\theta}_t^{\text{target}}$, $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}_t$, $\boldsymbol{\xi}^{\text{target}} \leftarrow \boldsymbol{\xi}_t^{\text{target}}$, and $\boldsymbol{\xi} \leftarrow \boldsymbol{\xi}_t$
return $\boldsymbol{\theta}^{\text{target}}$ and $\boldsymbol{\xi}^{\text{target}}$

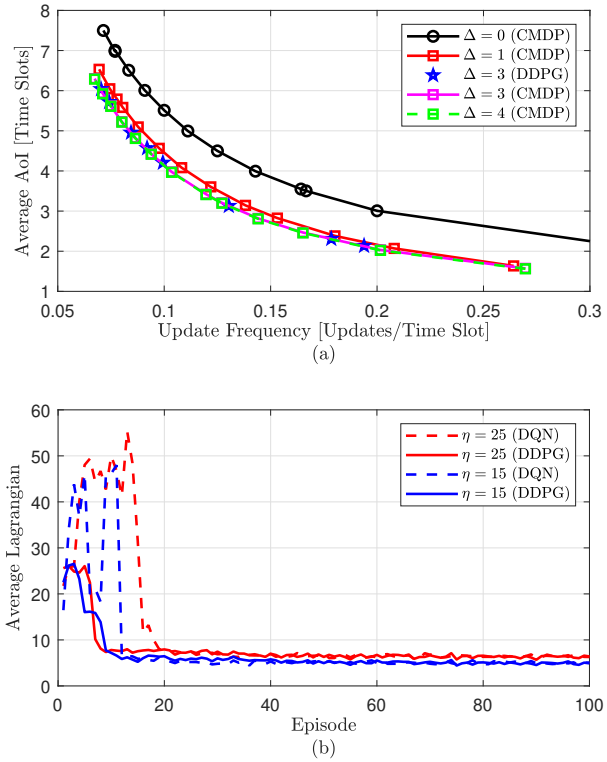


Fig. 3: (a) The average AoI obtained by CMDP and DDPG algorithms v.s. cache update frequency μ, as the window size Δ increases from 0 to 4. One dynamic content is being cached in the system. (b) Plot of the Lagrangian from subsequent episodes in the DDPG and DQN training.

where $0 < \rho < 1$. In Algorithm 3, we list the key steps of the algorithm we used to implement the DDPG algorithm.

IV. Performance Evaluation

In this section, we perform simulation studies to validate the analytical results. We compare the proposed optimal and suboptimal algorithms with the periodic update algorithm proposed in [24]. Unless specified otherwise, we set $T = 10,000$, $\hat{A} = 50$, $\Delta = 3$, $\eta = 100$, and $N_f = 3$, $\forall f \in \mathcal{F}$. The DDPG algorithm was implemented using PyTorch [14], and the parameters used for DDPG are as follows: $\rho = 0.01$, $\tau = 0.01$, $\beta_s = 0.001$, $N_{\text{epi}} = 200$, $T_{\text{epi}} = 1,000$, $T_{\text{update}} = 5,000$, $\sigma^2 = 0.0001$ and $K_{\text{batch}} = 32$.

A. Case 1: Single Dynamic Content

In Fig. 3 (a), we plot the average AoI obtained by the optimal policy, as the window size Δ increases from 0 to 4. We observe that under the same update frequency, the average AoI obtained by the optimal policy decreases with increasing window size. However, increasing the window size also increases the memory required to store the optimal policy. Since we observe that the average AoI obtained by the optimal policy with $\Delta = 4$ is close to the performance with $\Delta = 3$, in the remaining of this section, we will use $\Delta = 3$ as the default window size of the CMDP problem, due to its performance and complexity. The average AoI obtained by the DDPG algorithm is also shown in Fig. 3 (a), and we observe that the performance of the DDPG algorithm is close to the optimal solution of the CMDP problem. In Fig. 3 (b), we plot the convergence performance of the DDPG algorithm with two different values of η. We observe the average Lagrangian converged in all three cases after 20 episodes. Therefore, our choice of training the DDPG for 100 episodes is appropriate. The convergence performance of another DQN-based method [32], which does not exploit the structural property of the optimal policy, denoted as DQN in Fig. 3 (b), is also shown for comparison. We observe that our proposed DDPG algorithm has a better convergence performance.

In Fig. 4, we use the box and whisker plot[7] to compare the optimal policy of the CMDP problem with the offline optimal solution for a finite horizon problem, where all the user arrival requests and target download time are known *a priori*. The offline policy is obtained through exhaustive search. Due to the prohibitively high computational complexity of the exhaustive search method, we only consider a problem with small dimension, where $T = 50$, $F = 1$, and $N_f = 1$. We compare the optimal policy when $\Delta = 0$, 1, and 3 with the offline optimal policy. We observe that even for a small window size, *i.e.*, $\Delta = 3$, the average AoI obtained from the CMDP policy is close to the offline optimal solution.

[7]In a box and whisker plot, the upper and lower ends of the boxes are the upper and lower quartiles of the data samples. The median is marked by the horizontal line inside the box, while the two whiskers extend to the maximum and minimum values of the data samples.

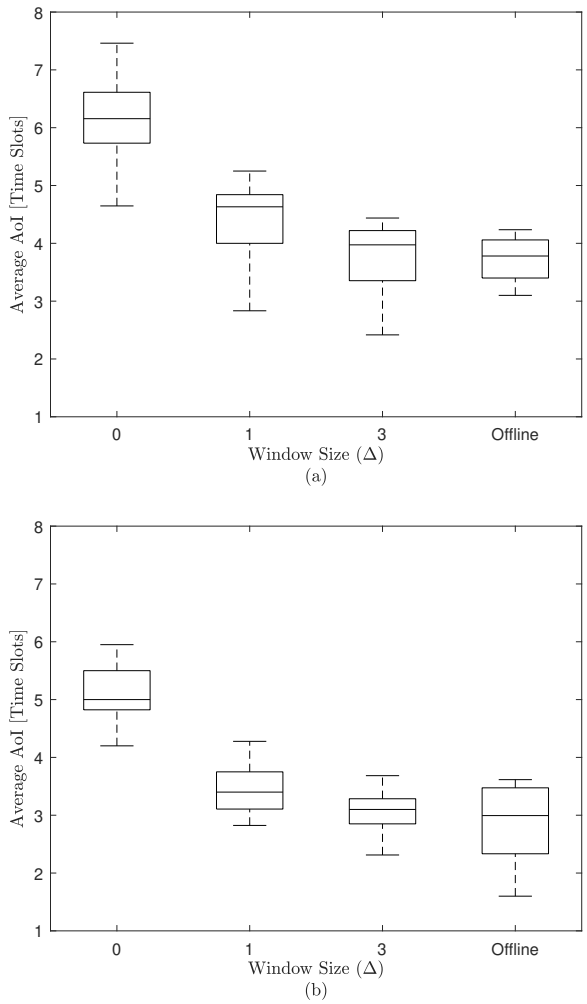


Fig. 4: Box and whisker plot of the average AoI obtained by the CMDP optimal policy and the offline optimal policy v.s. window size in a finite horizon problem where $T = 50$, and (a) four cache updates (*i.e.*, $\sum_{t=0}^{T-1} \mathbf{I}(u_t > 0) = 4$) and (b) five cache updates (*i.e.*, $\sum_{t=0}^{T-1} \mathbf{I}(u_t > 0) = 5$) are performed, respectively. 'Offline' corresponds to the offline optimal policy when all future user arrival requests are known *a priori*.

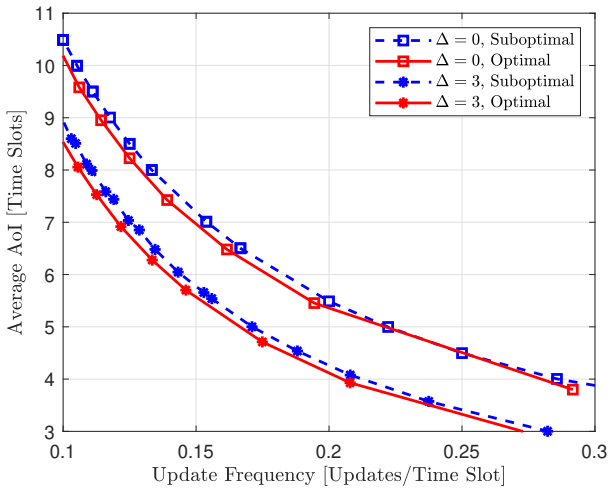


Fig. 5: Average AoI obtained by the optimal and suboptimal policies v.s. the frequency of cache update μ when two dynamic contents are being cached in the system. The window size Δ is equal to either 0 or 3.

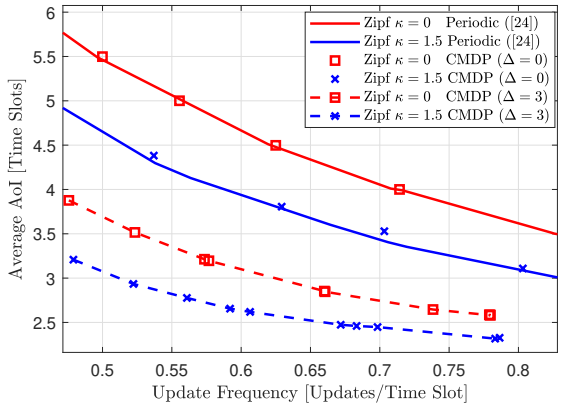


Fig. 6: Average AoI obtained by the proposed CMDP algorithm compared to the periodic update approach [24], under Zipf's distribution with parameter $\kappa = 0$ and $\kappa = 1.5$.

B. Case 2: Multiple Dynamic Contents

In Fig. 5, we plot the average AoI of two dynamic contents, for the case where the window size is set to either 0 or 3. The AoI obtained by using the optimal policy (in solid lines) is also included to evaluate the performance of the enforced decomposition algorithm (in dashed lines). We observe that there is only a small gap between the optimal and suboptimal algorithms. The average AoI performance is better when the window size Δ is equal to 3.

Next, we plot the average AoI obtained by the proposed CMDP-based algorithm, for $\Delta = 0$ and $\Delta = 3$, and compared it with the periodic update approach proposed in [24]. Five dynamic contents are being cached in the HetNet, where λ_f follows the Zipf's distribution with parameter κ. That is,

$$\lambda_f = \frac{f^{-\kappa}}{\sum_{i=1}^{F} i^{-\kappa}}, \quad f \in \mathcal{F}. \tag{49}$$

In Fig. 6, we plot the performance of the three algorithms when $\kappa = 0$ and $\kappa = 1.5$. From Fig. 6, we observe that the proposed CMDP algorithm when $\Delta = 0$ achieves similar AoI performance compared to the periodic update approach proposed in [24]. This is because when $\Delta = 0$, the CMDP algorithm updates the dynamic content f based on whether A_t^f exceeds a threshold ψ_t^f, which results in periodic update of f with an approximate period ψ_t^f. For both $\Delta = 0$ and $\Delta = 3$, the average AoI performance of the proposed CMDP algorithm improves when $\kappa > 0$.

In Fig. 7, we plot the average AoI performance obtained by the suboptimal policy obtained using the CMDP and DDPG algorithms when five and ten dynamic contents are being cached, and user request arrival rates follow the Zipf's distribution with parameter $\kappa = 0$. We observe that both the DDPG and CMDP algorithms achieve better AoI performance compared to the periodic update method. For example, when $F = 5$ and $\mu = 0.5$, the average AoI obtained by periodic update is approximately 6 time slots while the average AoI

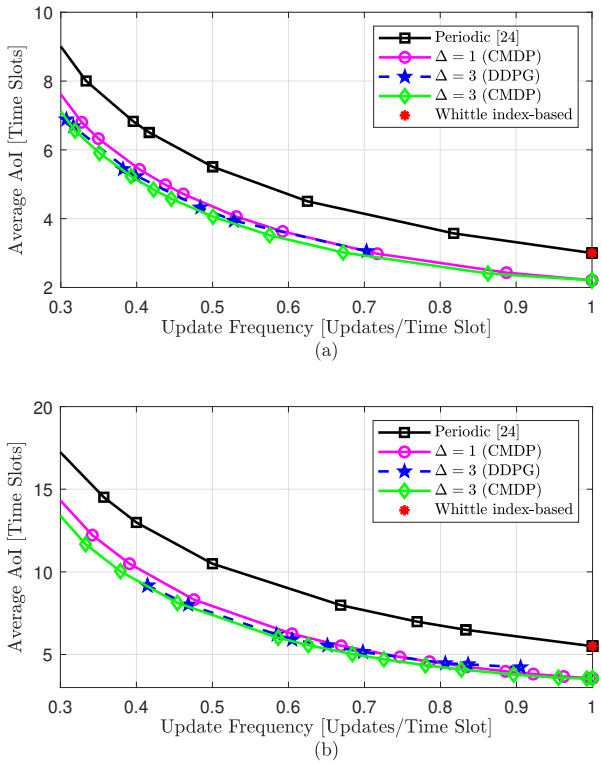


Fig. 7: Average AoI obtained by the CMDP, DDPG, and the periodic update algorithms v.s. cache update frequency μ, when (a) five dynamic contents are being cached and (b) ten dynamic contents are being cached. For $\mu = 1$ and $\Delta = 0$, the performance of the Whittle index-based algorithm is also shown.

obtained by the proposed algorithms are approximately 4 time slots, which resulted in a 30% improvement in the average AoI performance. The DDPG algorithm achieved comparable performance as the CMDP algorithm. When $\mu = 1$ and $\Delta = 0$, the problem can also be solved using the Whittle index-based scheduling algorithm introduced in [23], which is also denoted in Fig. 7. We observe that the CMDP algorithm matches the solution obtained by the Whittle index-based approach when $\Delta = 0$.

C. Validation of the Structural Results

Finally, in Fig. 8, we plot the optimal policy for different states when only one dynamic content is being cached, and different window sizes are used. The states that are not shown correspond to those states whose optimal policy is always not to update. In Fig. 8 (a), we observe that when $\Delta = 0$, the cache content update scheduling policy is only based on the current AoI of the served dynamic content in the current time slot, and is equivalent to the periodic updating policy. In Fig. 8 (b), we observe that when $\Delta = 1$, the optimal action depends on whether there is a user request that needs to be served in the next decision epoch. The optimal action is not to update the CS when no user request that needs to be served in the next decision epoch; and to only update the cache content when the current AoI is above a certain threshold, otherwise. In Figs. 8 (c) and (d), we observe that when Δ is equal to either 2 or 3, the optimal policy depends on $Q_t^{1,1}, \ldots, Q_t^{1,\Delta-1}$, G_t^1, and A_t^1. Nevertheless, in all the cases tested here, the optimal

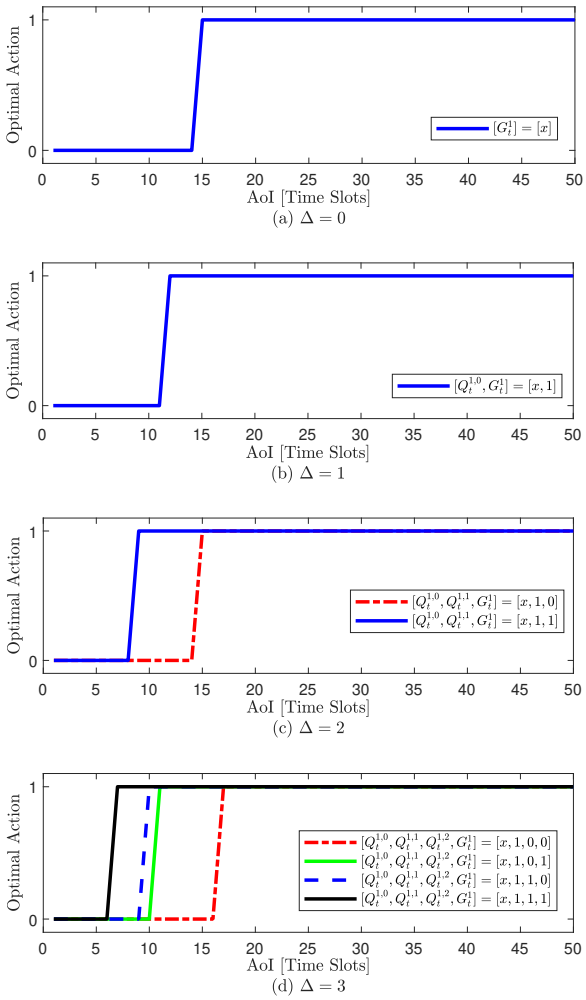


Fig. 8: Optimal policy for different states when only one dynamic content is being cached. The window size Δ is set to 0, 1, 2, and 3, respectively. The x in the legend means that the corresponding element can either be zero or one.

policy has a threshold structure with respect to the current AoI of the file.

V. Conclusion

In this paper, we studied the problem of caching dynamic contents using a HetNet architecture. We formulated the problem where the target download time for user requests in a short future time window is known, and designed a strategy where the scheduling decision depends on the user requests that need to be served in the near future. We formulated the problem as a CMDP. However, due to large state space, finding the optimal policy becomes computationally intractable. To this end, we designed a suboptimal algorithm that decomposes the original problem into separate subproblems, and determined a low-complexity suboptimal solution to the problem. We also proved that the optimal policy for caching a dynamic content has a threshold structure. To further reduce the memory required to store the optimal policy of the CMDP, we exploited the threshold structure and proposed a DRL framework based on DDPG to estimate the optimal policy of the CMDP. Simulation results show that both of the suboptimal algorithms

have near-optimal performance. The suboptimal algorithms outperform the periodic update scheme in different settings. Future research will consider more complex system dynamics, such as time varying channel conditions and the soft service deadline for different user requests.

APPENDIX

A. Proof of Lemma 3

Proof: We prove Lemma 3 using the RVIA algorithm (as shown in Algorithm 1) and mathematical induction. For readability, we will drop the superscript f and Lagrange multiplier η_f. From line 14 of Algorithm 1, we have for each state $\mathbf{s} \in \mathcal{S}_f$ and action $u \in \mathcal{U}_F$,

$$V_{k+1}(\mathbf{s}, u) = c(\mathbf{s}, u) + \sum_{\mathbf{s}_k \in \mathcal{S}_f} \mathbb{P}(\mathbf{s}_k \mid \mathbf{s}, u) J_k(\mathbf{s}_k). \tag{50}$$

According to lines 14 and 18 of Algorithm 1,

$$J_{k+1}(\mathbf{s}) = \min_{u \in \mathcal{U}_F} \; V_{k+1}(\mathbf{s}, u) - H_{k+1}(\mathbf{s}^{\text{ref}}), \tag{51}$$

where $\mathbf{s}^{\text{ref}}$ is a chosen reference state. According to [35], under any initialization of $J_0(\mathbf{s})$, the generated sequence $\{J_k(\mathbf{s})\}$ converges to $J(\mathbf{s})$, i.e.,

$$\lim_{k \to \infty} J_k(\mathbf{s}) = J(\mathbf{s}), \qquad \forall\ \mathbf{s} \in \mathcal{S}_f, \tag{52}$$

where $J(\mathbf{s})$ satisfies the Bellman equations (32). Let π_k^* denote the optimal policy at the k-th iteration, i.e.,

$$\pi_k^*(\mathbf{s}) = \arg\min_{u \in \mathcal{U}_F} V_k(\mathbf{s}, u), \qquad \forall\ \mathbf{s} \in \mathcal{S}_f. \tag{53}$$

Let us define two states $\mathbf{s},\ \mathbf{s}^+ \in \mathcal{S}_f$, where

$$\mathbf{s} = (A, Q^0, \ldots, Q^{\Delta-1}, G) \tag{54}$$

and

$$\mathbf{s}^+ = (A^+, Q^{0+}, \ldots, Q^{\Delta-1+}, G^+). \tag{55}$$

Suppose states $\mathbf{s}$ and $\mathbf{s}^+$ satisfy $A^+ \geq A$, $Q^\delta = Q^{\delta+}$, $\forall\ \delta \in \{0, \ldots, \Delta-1\}$ and $G = G^+$. We define this relationship as $\mathbf{s}^+ \succcurlyeq_A \mathbf{s}$. We aim to show that $J_k(\mathbf{s}^+) \geq J_k(\mathbf{s})$ holds for all $k = 1, 2, \ldots$. From (30), we have $c(\mathbf{s}, u) \leq c(\mathbf{s}^+, u)$, $\forall\ u \in \mathcal{U}_F,\ \mathbf{s}^+ \succcurlyeq_A \mathbf{s}$. Suppose the RVIA algorithm is used, and we initialize $J_0(\mathbf{s}) = 0$, $\forall\ \mathbf{s} \in \mathcal{S}_f$, we have $J_0(\mathbf{s}) \leq J_0(\mathbf{s}^+)$, $\forall\ \mathbf{s}^+ \succcurlyeq_A \mathbf{s}$. Now suppose $J_k(\mathbf{s}) \leq J_k(\mathbf{s}^+)$, $\forall\ \mathbf{s}^+ \succcurlyeq_A \mathbf{s}$, we would like to show $J_{k+1}(\mathbf{s}) \leq J_{k+1}(\mathbf{s}^+)$, $\forall\ \mathbf{s}^+ \succcurlyeq_A \mathbf{s}$. Since the optimal action in the $(k+1)$-th decision epoch is $\pi_{k+1}^*(\mathbf{s}^+)$, $\forall\ \mathbf{s}^+ \in \mathcal{S}_f$, we have

$$\begin{aligned}
&J_{k+1}(\mathbf{s}^+) + H_{k+1}(\mathbf{s}^{\text{ref}}) \\
&= c(\mathbf{s}^+, \pi_{k+1}^*(\mathbf{s}^+)) + \sum_{\mathbf{s}_k^+ \in \mathcal{S}_f} \mathbb{P}(\mathbf{s}_k^+ \mid \mathbf{s}^+, \pi_{k+1}^*(\mathbf{s}^+)) J_k(\mathbf{s}_k^+) \\
&= c(\mathbf{s}^+, \pi_{k+1}^*(\mathbf{s}^+)) \\
&\quad + \sum_{\mathbf{s}_k^+ \in \mathcal{S}_f} \mathbb{P}(\mathbf{s}_k^+ \mid \mathbf{s}^+, \pi_{k+1}^*(\mathbf{s}^+)) J_k(\mathbf{s}_k^+) \mathbf{I}(\pi_{k+1}^*(\mathbf{s}^+) = 1) \\
&\quad + \sum_{\mathbf{s}_k^+ \in \mathcal{S}_f} \mathbb{P}(\mathbf{s}_k^+ \mid \mathbf{s}^+, \pi_{k+1}^*(\mathbf{s}^+)) J_k(\mathbf{s}_k^+) \mathbf{I}(\pi_{k+1}^*(\mathbf{s}^+) = 0) \\
&\overset{(a)}{\geq} c(\mathbf{s}, \pi_{k+1}^*(\mathbf{s}^+)) \\
&\quad + \sum_{\mathbf{s}_k^+ \in \mathcal{S}_f} \mathbb{P}(\mathbf{s}_k^+ \mid \mathbf{s}, \pi_{k+1}^*(\mathbf{s}^+)) J_k(\mathbf{s}_k) \mathbf{I}(\pi_{k+1}^*(\mathbf{s}^+) = 1) \\
&\quad + \sum_{\mathbf{s}_k^+ \in \mathcal{S}_f} \mathbb{P}(\mathbf{s}_k^+ + \mathbf{s} - \mathbf{s}^+ \mid \mathbf{s}, \pi_{k+1}^*(\mathbf{s}^+)) \\
&\quad \times J_k(\mathbf{s}_k^+ + \mathbf{s} - \mathbf{s}^+) \mathbf{I}(\pi_{k+1}^*(\mathbf{s}^+) = 0) \\
&= c(\mathbf{s}, \pi_{k+1}^*(\mathbf{s}^+)) + \sum_{\mathbf{s}_k \in \mathcal{S}_f} \mathbb{P}(\mathbf{s}_k \mid \mathbf{s}, \pi_{k+1}^*(\mathbf{s}^+)) J_k(\mathbf{s}_k) \\
&\overset{(b)}{\geq} c(\mathbf{s}, \pi_{k+1}^*(\mathbf{s})) + \sum_{\mathbf{s}_k \in \mathcal{S}_f} \mathbb{P}(\mathbf{s}_k \mid \mathbf{s}, \pi_{k+1}^*(\mathbf{s})) J_k(\mathbf{s}_k) \\
&= J_{k+1}(\mathbf{s}) + H_{k+1}(\mathbf{s}^{\text{ref}}).
\end{aligned}$$

Note that (24), (30), and the induction assumption give inequality (a). Inequality (b) is due to the fact that $\pi_{k+1}^*(\mathbf{s})$ is the optimal action. The monotonicity of $J(\mathbf{s})$ in G and $Q^\delta,\ \forall\ \delta \in \{0, \ldots, \Delta-1\}$ can be proved in a similar manner. The monotonicity of $V(\mathbf{s}, u)$ follows directly from the monotonicity of $J(\mathbf{s})$ and (30). ■

B. Proof of Lemma 4

Proof: Similar to the proof of Lemma 3, we will drop the superscript f and Lagrange multiplier η_f. We prove that $\Gamma_{1,0}(\mathbf{s})$ is a monotonically non-increasing function in A, by using mathematical induction. We use the RVIA algorithm and define

$$\Gamma_{k,1,0}(\mathbf{s}) \triangleq V_k(\mathbf{s}, 1) - V_k(\mathbf{s}, 0), \qquad \mathbf{s} \in \mathcal{S}_f, \tag{56}$$

where $V_k(\mathbf{s}, u)$ represents the state-action value at the k-th iteration of RVIA. Since

$$\lim_{k \to \infty} J_k(\mathbf{s}) = J(\mathbf{s}), \qquad \mathbf{s} \in \mathcal{S}_f, \tag{57}$$

we also have

$$\lim_{k \to \infty} V_k(\mathbf{s}, u) = V(\mathbf{s}, u), \qquad \mathbf{s} \in \mathcal{S}_f, u \in \mathcal{U}_F. \tag{58}$$

Hence

$$\lim_{k \to \infty} \Gamma_{k,1,0}(\mathbf{s}) = \Gamma_{1,0}(\mathbf{s}), \qquad \mathbf{s} \in \mathcal{S}_f. \tag{59}$$

Consider two state vectors $\mathbf{s}^+$ and $\mathbf{s}$ defined in (54) and (55), where $\mathbf{s}^+ \succcurlyeq_A \mathbf{s}$, we would like to show that

$$\Gamma_{k,1,0}(\mathbf{s}^+) \leq \Gamma_{k,1,0}(\mathbf{s}) \tag{60}$$

holds for all $k = 1, 2, \ldots$. We have

$$\begin{aligned}
&\Gamma_{k,1,0}(\mathbf{s}^+) - \Gamma_{k,1,0}(\mathbf{s}) \\
&= V_k(\mathbf{s}^+, 1) - V_k(\mathbf{s}, 1) - V_k(\mathbf{s}^+, 0) + V_k(\mathbf{s}, 0) \\
&= c(\mathbf{s}^+, 1) + \sum_{\mathbf{s}_k^+ \in \mathcal{S}_f} \mathbb{P}(\mathbf{s}_k^+ \mid \mathbf{s}^+, 1) J_k(\mathbf{s}_k^+) \\
&\quad - c(\mathbf{s}, 1) - \sum_{\mathbf{s}_k \in \mathcal{S}_f} \mathbb{P}(\mathbf{s}_k \mid \mathbf{s}, 1) J_k(\mathbf{s}_k) \\
&\quad - c(\mathbf{s}^+, 0) - \sum_{\mathbf{s}_k^+ \in \mathcal{S}_f} \mathbb{P}(\mathbf{s}_k^+ \mid \mathbf{s}^+, 0) J_k(\mathbf{s}_k^+) \\
&\quad + c(\mathbf{s}, 0) + \sum_{\mathbf{s}_k \in \mathcal{S}_f} \mathbb{P}(\mathbf{s}_k \mid \mathbf{s}, 0) J_k(\mathbf{s}_k)
\end{aligned}$$

$$\begin{aligned}
&\overset{(a)}{=} \sum_{\mathbf{s}_k^+ \in \mathcal{S}_f} \mathbb{P}(\mathbf{s}_k^+ \mid \mathbf{s}^+, 1) J_k(\mathbf{s}_k^+) - \sum_{\mathbf{s}_k \in \mathcal{S}_f} \mathbb{P}(\mathbf{s}_k \mid \mathbf{s}, 1) J_k(\mathbf{s}_k) \\
&\quad - \sum_{\mathbf{s}_k^+ \in \mathcal{S}_f} \mathbb{P}(\mathbf{s}_k^+ \mid \mathbf{s}^+, 0) J_k(\mathbf{s}_k^+) + \sum_{\mathbf{s}_k \in \mathcal{S}_f} \mathbb{P}(\mathbf{s}_k \mid \mathbf{s}, 0) J_k(\mathbf{s}_k) \\
&\overset{(b)}{\leq} - \sum_{\mathbf{s}_k^+ \in \mathcal{S}_f} \mathbb{P}(\mathbf{s}_k^+ \mid \mathbf{s}^+, 0) J_k(\mathbf{s}_k^+) + \sum_{\mathbf{s}_k \in \mathcal{S}_f} \mathbb{P}(\mathbf{s}_k \mid \mathbf{s}, 0) J_k(\mathbf{s}_k) \\
&= - \sum_{\mathbf{s}_k^+ \in \mathcal{S}_f} \mathbb{P}(\mathbf{s}_k^+ \mid \mathbf{s}^+, 0) \Big(J_k(\mathbf{s}_k^+) - J_k(\mathbf{s}_k^+ - \mathbf{s}^+ + \mathbf{s}) \Big) \\
&\overset{(c)}{\leq} 0.
\end{aligned}$$

Equality (a) comes from (30), which leads to

$$c(\mathbf{s}^+, 1) - c(\mathbf{s}^+, 0) = c(\mathbf{s}, 1) - c(\mathbf{s}, 0) = \eta_f. \tag{61}$$

Since the AoI of the dynamic content is reset to 1 after a cache content update action is chosen, we have $\mathbb{P}(A_k \mid \mathbf{s}, 1) = \mathbb{P}(A_k \mid \mathbf{s}^+, 1) = \mathbf{I}(A_k = 1)$. Hence, according to (24), we have $\mathbb{P}(\mathbf{s}_k \mid \mathbf{s}, 1) = \mathbb{P}(\mathbf{s}_k \mid \mathbf{s}^+, 1)$, $\forall\ s^+ \succcurlyeq_A s$, which leads to inequality (b). Inequality (c) is due to the monotonicity of J_k from Lemma 3 and that $\mathbf{s}_k^+ \succcurlyeq_A \ \mathbf{s}_k^+ - \mathbf{s}^+ + \mathbf{s}$. ■

C. Proof of Theorem 1

Proof: Given two states $\mathbf{s}^+$ and $\mathbf{s}$ defined in (54) and (55), where $s^+ \succcurlyeq_A s$, due to the monotonicity of $\Gamma_{1,0}^f(\mathbf{s}_t^f, \eta_f)$ from Lemma 4, we have

$$\begin{aligned}
&V^f(\mathbf{s}^+, 1, \eta_f) - V^f(\mathbf{s}^+, 0, \eta_f) \\
&\leq V^f(\mathbf{s}, 1, \eta_f) - V^f(\mathbf{s}, 0, \eta_f).
\end{aligned} \tag{62}$$

Therefore, if action 1 dominates in state $\mathbf{s}$, then it will also dominate in state $\mathbf{s}^+$. ■

REFERENCES

[1] M. Ma and V. W. S. Wong, "A deep reinforcement learning approach for dynamic contents caching in HetNets," in *Proc. of IEEE Int'l Conf. on Commun. (ICC)*, Jun. 2020.

[2] K. Shanmugam, N. Golrezaei, A. G. Dimakis, A. F. Molisch, and G. Caire, "FemtoCaching: Wireless content delivery through distributed caching helpers," *IEEE Trans. Inf. Theory*, vol. 59, no. 12, pp. 8402–8413, Dec. 2013.

[3] P. Blasco and D. Gunduz, "Learning-based optimization of cache content in a small cell base station," in *Proc. of IEEE Int'l Conf. on Commun. (ICC)*, Sydney, Australia, Jun. 2014.

[4] T. Hou, G. Feng, S. Qin, and W. Jiang, "Proactive content caching by exploiting transfer learning for mobile edge computing," in *Proc. of IEEE Global Commun. Conf. (GLOBECOM)*, Singapore, Dec. 2017.

[5] M. A. Maddah-Ali and U. Niesen, "Fundamental limits of caching," *IEEE Trans. Inf. Theory*, vol. 60, no. 5, pp. 2856–2867, May 2014.

[6] L. Xiang, D. W. K. Ng, R. Schober, and V. W. S. Wong, "Cache-enabled physical layer security for video streaming in backhaul-limited cellular networks," *IEEE Trans. Wireless Commun.*, vol. 17, no. 2, pp. 736–751, Feb. 2018.

[7] B. Zhou, Y. Cui, and M. Tao, "Stochastic content-centric multicast scheduling for cache-enabled heterogeneous cellular networks," *IEEE Trans. Wireless Commun.*, vol. 15, no. 9, pp. 6284–6297, Sep. 2016.

[8] B. Zhou, Y. Cui, and M. Tao, "Optimal dynamic multicast scheduling for cache-enabled content-centric wireless networks," *IEEE Trans. Commun.*, vol. 65, no. 7, pp. 2956–2970, Jul. 2017.

[9] L. Xiang, D. W. K. Ng, X. Ge, Z. Ding, V. W. S. Wong, and R. Schober, "Cache-aided non-orthogonal multiple access: The two-user case," *IEEE Trans. Sel. Areas. Signal Process.*, vol. 13, no. 3, pp. 436–451, Jun. 2019.

[10] M. Ma and V. W. S. Wong, "An optimal peak hour content server cache update scheduling algorithm for 5G HetNets," in *Proc. of IEEE Int'l Conf. on Commun. (ICC)*, Shanghai, China, May 2019.

[11] V. W. S. Wong, R. Schober, D. W. K. Ng, and L. Wang, *Key Technologies for 5G Wireless Systems*. Cambridge University Press, 2017.

[12] M. S. Ali, M. Vecchio, M. Pincheira, K. Dolui, F. Antonelli, and M. Rehmani, "Applications of blockchains in the Internet of things: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 2, pp. 1676–1717, Second Quarter 2019.

[13] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[14] PyTorch. (2020) torchvision.models. [Online]. Available: https://pytorch.org/docs/stable/torchvision/models.html

[15] Google. (2020) Introduction to TensorFlow lite. [Online]. Available: https://www.tensorflow.org/lite/overview

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

[17] G. Sartoretti, Y. Wu, W. Paivine, T. S. Kumar, S. Koenig, and H. Choset, "Distributed reinforcement learning for multi-robot decentralized collective construction," in *Distributed Autonomous Robotic Systems*. Springer, 2019, pp. 35–49.

[18] S. Kaul, R. Yates, and M. Gruteser, "Real-time status: How often should one update?" in *Proc. IEEE Int'l Conf. on Computer Commun. (INFOCOM) Mini-Conf.*, Orlando, FL, Mar 2012.

[19] 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.

[20] I. Kadota, A. Sinha, and E. Modiano, "Optimizing age of information in wireless networks with throughput constraints," in *Proc. IEEE Int'l Conf. on Computer Commun. (INFOCOM)*, Honolulu, HI, Apr. 2018.

[21] Z. Jiang, B. Krishnamachari, S. Zhou, and Z. Niu, "Can decentralized status update achieve universally near-optimal age-of-information in wireless multiaccess channels?" in *Proc. Int'l Teletraffic Congress*, Vienna, Austria, Sep. 2018.

[22] I. Kadota, E. Uysal-Biyikoglu, R. Singh, and E. Modiano, "Minimizing the age of information in broadcast wireless networks," in *Proc. Annual Allerton Conf. on Communi., Control, and Computing*, Monticello, IL, Sep. 2016.

[23] Y. Hsu, "Age of information: Whittle index for scheduling stochastic arrivals," in *Proc. IEEE Int'l Symp. Information Theory (ISIT)*, Vail, CO, Jun. 2018.

[24] R. D. Yates, P. Ciblat, A. Yener, and M. Wigger, "Age-optimal constrained cache updating," in *Proc. IEEE Int'l Symp. on Inf. Theory (ISIT)*, Aachen, Germany, Jun. 2017.

[25] H. Tang, P. Ciblat, J. Wang, M. Wigger, and R. Yates, "Age of information aware cache updating with file-and age-dependent update durations," *arXiv preprint arXiv:1909.05930*, 2019.

[26] 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," *arXiv preprint arXiv:2003.00383*, 2020.

[27] 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.

[28] ——, "Minimum age of information in the Internet of things with non-uniform status packet sizes," *IEEE Trans. Commun.*, vol. 19, no. 3, pp. 1933–1947, Mar. 2020.

[29] A. M. Bedewy, Y. Sun, S. Kompella, and N. B. Shroff, "Age-optimal sampling and transmission scheduling in multi-source systems," in *Proc. ACM Int'l Symp. Mobile Ad Hoc Networking and Computing (MobiHoc)*, Catania, Italy, Jul. 2019.

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

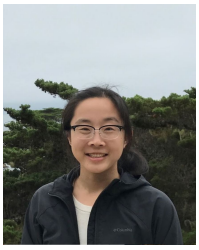
[31] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," in *Proc. Int'l Conf. on Learning Representations (ICLR)*, San Juan, Puerto Rico, May 2016.

[32] V. Mnih, K. Kavukcuoglu, D. Silver, A. Rusu, J. Veness, M. Bellemare, A. Graves, M. Riedmiller, A. Fidjeland *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

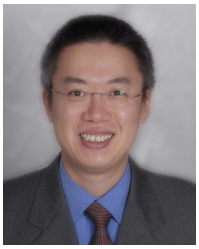
[33] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" in *Proc. of Advances in Neural Information Processing Systems Conf. (NIPS)*, Montreal, Canada, Dec. 2014.

[34] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int'l Conf. on Learning Representations (ICLR)*, San Diego, CA, May 2015.

[35] D. P. Bertsekas, *Dynamic Programming and Optimal Control, 4th Edition, Vol. I & II*. Athena Scientific, 2017.
[36] F. J. Beutler and K. W. Ross, "Optimal policies for controlled Markov chains with a constraint," *Journal of Mathematical Analysis and Applications*, vol. 112, no. 1, pp. 236–252, Nov. 1985.
[37] L. I. Sennott, "Constrained average cost Markov decision chains," *Probability in the Engineering and Informational Sciences*, vol. 7, no. 1, p. 69–83, Jan. 1993.
[38] H. Robbins and S. Monro, "A stochastic approximation method," *Ann. Math. Stat.*, vol. 22, no. 1, pp. 400–407, 1951.



Manyou Ma (S'19) received the B.Eng degree from McGill University, Montreal, Canada, in 2014 and the M.A.Sc. degree from the University of British Columbia (UBC), Vancouver, Canada, in 2016. She is currently pursuing the Ph.D. degree at UBC. She is a recipient of the UBC Four Year Doctoral Fellowship (4YF), Mitacs Accelerate Fellowship, and the Natural Sciences and Engineering Research Council of Canada (NSERC) Postgraduate Scholarship. Her research interests include deep learning-driven scheduling algorithms design, content caching in heterogeneous networks, and age of information.



Vincent W.S. Wong (S'94, M'00, SM'07, F'16) received the B.Sc. degree from the University of Manitoba, Winnipeg, MB, Canada, in 1994, the M.A.Sc. degree from the University of Waterloo, Waterloo, ON, Canada, in 1996, and the Ph.D. degree from the University of British Columbia (UBC), Vancouver, BC, Canada, in 2000. From 2000 to 2001, he worked as a systems engineer at PMC-Sierra Inc. (now Microchip Technology Inc.). He joined the Department of Electrical and Computer Engineering at UBC in 2002 and is currently a Professor. His research areas include protocol design, optimization, and resource management of communication networks, with applications to wireless networks, smart grid, mobile edge computing, and Internet of Things. Currently, Dr. Wong is an Executive Editorial Committee Member of *IEEE Transactions on Wireless Communications*, an Area Editor of *IEEE Transactions on Communications* and *IEEE Open Journal of the Communications Society*, and an Associate Editor of *IEEE Transactions on Mobile Computing*. He is a Technical Program Co-chair of the *IEEE* 92*nd Vehicular Technology Conference (VTC*2020*-Fall)*. He has served as a Guest Editor of *IEEE Journal on Selected Areas in Communications* and *IEEE Wireless Communications*. He has also served on the editorial boards of *IEEE Transactions on Vehicular Technology* and *Journal of Communications and Networks*. He was a Tutorial Co-chair of *IEEE Globecom*'18, a Technical Program Co-chair of *IEEE SmartGridComm*'14, as well as a Symposium Co-chair of *IEEE ICC*'18, *IEEE SmartGridComm* ('13, '17) and *IEEE Globecom*'13. He is the Chair of the IEEE Vancouver Joint Communications Chapter and has served as the Chair of the IEEE Communications Society Emerging Technical Sub-Committee on Smart Grid Communications. He is an IEEE Communications Society Distinguished Lecturer (2019−2020).