Age-Aware Edge Caching and Multicast Scheduling Using Deep Reinforcement Learning

Seyedeh Bahereh Hassanpour*†, Ahmad Khonsari*†, Masoumeh Moradian†, Aresh Dadlani‡§, and Galymzhan Nauryzbayev§

*School of Electrical and Computer Engineering, College of Engineering, University of Tehran, Tehran, Iran
†School of Computer Science, Institute for Research in Fundamental Sciences (IPM), Tehran, Iran
‡Department of Computing Science, University of Alberta, Edmonton, Canada
§Department of Electrical and Computer Engineering, Nazarbayev University, Astana, Kazakhstan
Emails: a_khonsari@ut.ac.ir, {bahereh, mmoradian}@ipm.ir, {aresh.dadlani, galymzhan.nauryzbayev}@nu.edu.kz

***Abstract*—The temporal nature of data in Internet of Things (IoT) networks necessitates periodic updates of cached content at edge devices, while multicasting dynamic content can enhance network efficiency. This paper addresses the challenge of joint cache updating and multicast scheduling in a cache-enabled, queue-equipped small base station (SBS) with limited cache capacity, which accesses a macro base station (MBS) to download (update) uncached (cached) content and serves requests through multicasting. We formulate a two-stage optimization problem to minimize the average age of information (AAoI) per request, subject to constrained average queueing delay and access rate. The first stage employs the Lyapunov drift-plus-penalty method at the SBS to schedule multicasting and downloading (updating) uncached (cached) content. The second stage, implemented at the MBS, leverages deep reinforcement learning (DRL) to determine the content replacement policy. Simulation results show that the DRL-based cache replacement policy yields up to 50%, 59%, and 60% improvements in AAoI compared to the maximum age, least-recently-used, and least-frequently-used baseline policies, respectively.**

***Index Terms*—Age of information, edge cache updating, Lyapunov optimization, deep reinforcement learning, multicasting.**

I. Introduction

Edge caching is widely recognized as a pivotal strategy for managing the increasing traffic demands from smart devices and mobile Internet services in emerging wireless networks. During low-traffic periods, popular content is cached at the edge and accessed during peak hours [1], [2]. However, the proliferation of real-time applications such as traffic reports and news feeds has necessitated more dynamic content handling. To ensure users receive up-to-date content and avoid serving stale data, timely updates to cached content are imperative. The freshness of content is typically quantified using the age of information (AoI) metric, which measures the time elapsed since the latest version of the content was generated [3].

Edge caching networks face two notable limitations. Firstly, the access rate to a global library containing fresh content, such as a macro base station (MBS), is limited, which directly affects content freshness. Secondly, the finite cache capacity at local caches, such as a small base station (SBS), requires careful decisions regarding the utilization of access opportunities, either for downloading uncached content or updating aged cached content. Several studies have explored AoI-aware cache updating policies under such constraints. To minimize the average AoI (AAoI) in a single cache with limited updating rates, the authors of [4] prove that the updating frequency of each content should be proportional to the square root of its popularity. The same problem is addressed in [5], where update intervals are assumed to depend on the files and their age. In contrast, the authors of [6] introduce stochastic content refreshing for the global library. In [7], the authors propose a Markov decision process (MDP)-based solution to minimize cache age, considering updating costs while ensuring timely user service. Under limited cache capacity, the authors of [8] reveal that caching decisions hinge on content popularity, while eviction times are linked to content age.

The above efforts ([4]–[8]) assume the instantaneous serving of requests for different contents without accounting for transmission delays. However, content delivery consumes both time and bandwidth resources of the local cache. Studies such as [9]–[11] address transmission delays by buffering requests for various contents in single or multiple queues. Nonetheless, in practical scenarios, multiple requests for the same content may arrive at the local cache, allowing concurrent servicing by multicasting the associated content. This reduces overall latency and delivers fresher content to end-users. For instance, in a streaming platform where multiple users request to watch the same movie or TV show, the local cache can efficiently serve the content to all users simultaneously, eliminating the need for multiple server accesses and reducing buffering time. Multicasting is particularly valuable for dynamic-content group delivery [12] and is also employed to ensure user privacy in status-updating networks [13]. In [14]–[16], the authors explore multicast scheduling for cached networks with static content to minimize the average delay and power costs of cellular networks with and without cache replacement. However, to the best of our knowledge, the efficacy of age-aware multicast scheduling in cache-limited edge networks remains unexplored.

To address this research gap, we examine an edge caching network consisting of an MBS and a cache-enabled SBS catering to users' requests. Our analysis incorporates various constraints, including the MBS's limited access rate, the SBS's cache capacity, and bounded average waiting times for requests. We propose a two-stage *age-aware* strategy to optimize multicasting, content

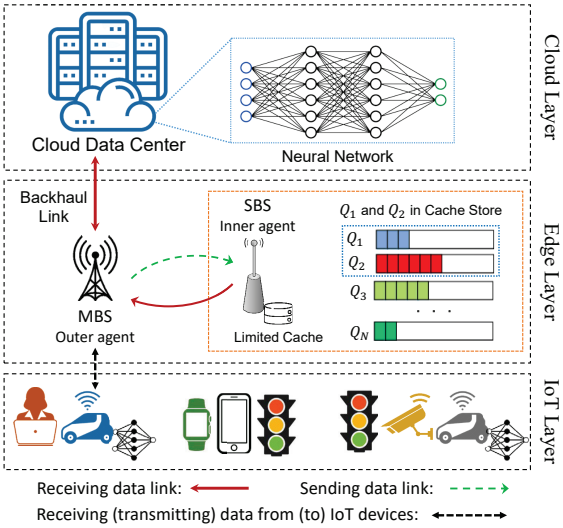


Fig. 1. Schema of the system model where the SBS decides on which request queue to serve/update and the MBS deals with the cache content replacement.

updating, and replacement. In the initial stage, we deploy a lightweight optimization approach driven by Lyapunov drift-plus-penalty criteria at the SBS to schedule multicast transmissions and make content update and download decisions. Subsequently, content replacement decisions are made at the MBS using deep reinforcement learning (DRL), leveraging information from the SBS. This stage is implemented at the MBS since, in many real-world scenarios, the MBS oversees cache content placement at SBSs. Simulation results corroborate the superiority of our DRL-based cache replacement method over existing policies such as maximum age (MA), least-frequently-used (LFU), and least-recently-used (LRU).

The rest of the paper is organized as follows. Section II presents the system model and problem formulation. The two stages of the proposed optimization problem are detailed in Sections III and IV, respectively. Simulation results are discussed in Section V. Finally, conclusions are drawn in Section VI.

II. System Model Description

We consider a single-cell edge caching network comprising one MBS and one SBS, as depicted in Fig. 1. The network contains a content library $\mathcal{C} = \{1, 2, \ldots, N\}$ with N fresh contents, all accessible by the MBS. The SBS is equipped with a cache capable of storing M ($M < N$) contents. At the outset of each time slot t, we designate $\mathcal{C}_t$ and $\bar{\mathcal{C}}_t$ as the sets of cached and uncached content indices, respectively, such that $|\mathcal{C}_t| = M$ and $|\bar{\mathcal{C}}_t| = N - M$. The SBS, responsible for handling requests, manages N queues $\mathcal{Q} = \{Q_i\}$, each containing user requests for content $i \in \mathcal{C}$. During each time slot, the SBS can multicast the requested content from only one queue, thereby emptying the respective queue. Additionally, in each time slot, it can either update one cached content or download one uncached content, with the downloaded content potentially replacing a cached one or being discarded. In what follows, we characterize the age of the contents and the dynamics of the queues and the cache based on the aforementioned decisions. Table 1 summarizes the key notations used in this section.

TABLE I
Summary of System Notations

Notation	Definition
$\mathcal{C}$	Library with N content files.
$\mathcal{C}_t$ ($\bar{\mathcal{C}}_t$)	Index set of cached (uncached) contents in the SBS cache of size M at time t.
$\mathcal{Q}=\{Q_i\}, \forall i \in \mathcal{C}$	User request queues, each of capacity B_i.
$\mathcal{A}_t = \{A_i^t\}, \forall i \in \mathcal{C}_t$	Age set of cached contents at time t.
A'^t_i	Age of content i multicast at time t.
α_i^t (μ_i^t)	Decision r.v. for updating/downloading (multicasting) content i at time t.
q_i^t	Number of pending requests in Q_i at time t.
λ_i^t	Number of requests entering Q_i at time t.
γ_i^t	Action taken with respect to content i at time t.
$\bar{A}_w$	Weighted AoI.

Since the SBS can update or download only one content per time slot, it follows that the contents are not consistently fresh at all times. Hence, we define the set $\mathcal{A}_t = \{A_i^t\}_{i \in \mathcal{C}_t}$ to contain the ages of the cached content at the onset of time slot t. Furthermore, let the binary random variable (b.r.v.) α_i^t indicate whether or not content i has been updated (or downloaded) for the cases $i \in \mathcal{C}_t$ or $i \in \bar{\mathcal{C}}_t$, respectively. As a result of this, we have $\sum_{i=1}^{N} \alpha_i^t \leq 1$, and for $i \in \mathcal{C}_t$, A_i^t evolves as:

$$A_i^{t+1} = A_i^t \left(1 - \alpha_i^t\right) + 1. \tag{1}$$

Evidently, (1) implies that A_i^t increases linearly if $\alpha_i^t = 0$. We define the b.r.v. μ_i^t to indicate whether content i is multicast during time slot t, such that $\sum_{i=1}^{N} \mu_i^t = 1$. Also, let q_i^t denote the number of pending requests in Q_i at the beginning of time slot t. Thus, q_i^{t+1} can be expressed as:

$$q_i^{t+1} = \min\left\{q_i^t \left(1 - \mu_i^t\right) + \lambda_i^t, B_i\right\}, \tag{2}$$

where λ_i^t is the number of requests entering Q_i at time slot t and B_i is the size of Q_i. We assume B_i to be sufficiently large to prevent any request dropouts. When $\mu_i^t = 1$, we define A'^t_i as the age of content i received by all pending requests in Q_i at the end of time slot t, which can be formulated as:

$$A'^t_i = \begin{cases} A_i^t\,(1 - \alpha_i^t) + 1; & \text{if } i \in \mathcal{C}_t, \\ 1; & \text{if } i \in \bar{\mathcal{C}}_t. \end{cases} \tag{3}$$

In (3), we assume that updating is completed prior to multicasting, and it takes significantly less time compared to the duration of a time slot. Moreover, the age of content delivered to the users, regardless of its freshness, increases by one unit since one slot duration is required for each transmission. Upon downloading uncached content from the MBS, it may replace one of the existing contents at the end of the time slot t. To formulate this process, let $\gamma_i^t \in \mathcal{C}_t \cup \{0\}$ denote the action to be taken in case of $\alpha_i^t = 1$ and $i \in \bar{\mathcal{C}}_t$. The value of $\gamma_i^t = 0$ implies that content i is discarded after being multicast, while $\gamma_i^t = j$ indicates that

content i will replace the existing content $j \in \mathcal{C}_t$. Consequently, the evolution of the content set $\mathcal{C}_t$ can be characterized as:

$$\mathcal{C}_{t+1} = \begin{cases} \{\mathcal{C}_t \setminus j\} \cup \{i\}; & \text{if } \mu_i^t = 1, i \in \bar{\mathcal{C}}_t, \gamma_i^t = j, \\ \mathcal{C}_t; & \text{if otherwise.} \end{cases} \tag{4}$$

Note that for the newly cached content, A_i^{t+1} takes the value of one, while for the rest of the contents, it is calculated using (1). Given this setting, we aim to optimize the variables $\{\mu_i^t, \alpha_i^t, \gamma_i^t\}$ over all time slots to minimize the average age of delivered contents per request, which we denote by $\bar{A}$. Thus, the above objective can be formally expressed as:

$$\bar{A} = \lim_{T\to\infty} \frac{\sum_{t=0}^{T}\sum_{i=1}^{N} q_i^t\, A'^t_i\, \mu_i^t}{\sum_{t=0}^{T}\sum_{i=1}^{N} q_i^t \mu_i^t}. \tag{5}$$

Assuming the stability and sufficient size of the queues to prevent packet drops, the denominator of (5) can be reduced to λT for sufficiently large values of T, where λ denotes the aggregate arrival rate of requests to the SBS ($\sum_{i=1}^{N} \lambda_i$), and is taken to be an unknown constant. Therefore, minimizing $\bar{A}$ is equivalent to minimizing the weighted AAoI, denoted by $\bar{A}_w$, and is given as:

$$\bar{A}_w = \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T}\sum_{i=1}^{N} q_i^t\, A'^t_i\, \mu_i^t. \tag{6}$$

We now impose constraints on the update rate at the SBS and the average delay of pending requests in Q_i. The former is upper bounded by some constant D as below:

$$\lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T}\sum_{i=1}^{N} \alpha_i^t \le D. \tag{7}$$

Using Little's law, constraining the latter translates to setting an upper bound on the average queue length as shown below, where d_i denotes the average waiting time threshold in the request queue for content i:

$$\lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T} q_i^t \le d_i, \quad \forall i \in \mathcal{C}. \tag{8}$$

In the subsequent sections, we propose a solution to minimize (6) subject to (7) and (8) using two decision-making agents. The first agent, referred to as the *inner agent*, is deployed at the SBS and leverages Lyapunov optimization to determine the optimal values of $\{\mu_i^t, \alpha_i^t\}$. The second agent, called the *outer agent*, resides at the MBS, and based on the feedback received from the inner agent, employs DRL to compute the optimal $\{\gamma_i^t\}$ values. Put simply, in each time slot, the inner agent chooses the queue to serve and the content to update or download. If downloading occurs, the outer agent specifies whether the downloaded content should replace a particular cached content or be discarded.

III. Inner Agent: Lyapunov Optimization

In this section, we focus on the Lyapunov drift-plus-penalty-based online policy adopted by the inner agent to jointly schedule content multicasting and updating/downloading. The inner agent receives $\mathcal{C}_t$, $\mathcal{A}_t$, and $\{q_i^t\}$ in time slot t, and finds the optimal $\{\mu_i^t, \alpha_i^t\}$ values that minimize $\bar{A}_w$ with respect to (7) and (8). To introduce the Lyapunov drift, we first define virtual queues Z_u and Z_i, $\forall i \in \mathcal{C}$, corresponding to constraints (7) and (8) as follows, where z_u^t and z_i^t denote, respectively, the lengths of Z_u and Z_i at the onset of time slot t:

$$z_u^{t+1} = \max\{z_u^t - D, 0\} + \sum_{i=1}^{N} \alpha_i^t, \tag{9}$$

$$z_i^{t+1} = \max\{z_i^t - d_i, 0\} + q_i^{t+1}. \tag{10}$$

Note that stabilizing the queues in (9) and (10) leads to satisfying constraints (7) and (8), respectively. By denoting $\mathcal{Z}_t = \{z_u^t, \{z_i^t\}\}$ to be the status of virtual queues, the Lyapunov function and Lyapunov drift, $L(t)$ and $\Delta L(t)$, can be defined as:

$$L(t) \triangleq \frac{1}{2}\sum_{i=1}^{N} (z_i^t)^2 + (z_u^t)^2, \tag{11}$$

$$\Delta L(t) \triangleq \mathbb{E}\left[L(t+1) - L(t) | \mathcal{Z}_t, \mathcal{C}_t, \mathcal{A}_t\right]. \tag{12}$$

On the other hand, the drift-plus-penalty criteria is defined as follows, where $V > 0$ is a chosen constant:

$$\Theta(t) = \Delta L(t) + V \sum_{i=1}^{N} q_i^t\, A'^t_i\, \mu_i^t. \tag{13}$$

Minimizing the drift and penalty terms in (13) leads to serving longer queues with less number of updates and shorter queues with lower ages, respectively, which signifies the trade-off between the queueing delay and AoI. Using the Lyapunov theorem in [17], the upper bound on $\Theta(t)$ can be obtained as:

$$\begin{aligned}\Theta(t) \le B - z_u^t D - \sum_{i=1}^{N} z_i^t\big(d_i - q_i^t - \lambda_i^t\big) - \sum_{i=1}^{N} z_i^t\, q_i^t\, \mu_i^t \\ + z_u^t \sum_{i=1}^{N} \alpha_i^t + V \sum_{i=1}^{N} q_i^t\, A'^t_i\, \mu_i^t,\end{aligned} \tag{14}$$

where $B = \frac{1}{2}\sum_{i=1}^{N}(d_i^2 + B_i^2) + \frac{1}{2}(D^2+1)$. Rather than optimize $\Theta(t)$ directly, we minimize the upper bound given in (14). Doing so minimizes the last three terms in (14) that are functions of μ_i^t and α_i^t. Thus, the cost function $\psi(t)$ is given as:

$$\psi(t) = \sum_{i=1}^{N} q_i^t\, \mu_i^t \big(V A'^t_i - z_i^t\big) + z_u^t\, \alpha_i^t. \tag{15}$$

The online policy of the inner agent, used in Algorithm 1 for decision-making, is introduced in Proposition 1 below.

Proposition 1. *Define the subsets* $\mathcal{S}_u^t \triangleq \{i \in \mathcal{C}_t | z_u^t < V q_i^t\, A_i^t\}$ *and* $\bar{\mathcal{S}}_u^t \triangleq \mathcal{C}_t \setminus \mathcal{S}_u^t$. *Also, derive content indices* l *and* m *as* $l = \arg\min_{i \in \mathcal{S}_u^t \cup \bar{\mathcal{C}}_t} \{q_i^t (V - z_i^t)\}$ *and* $m = \arg\min_{i \in \bar{\mathcal{S}}_u^t} \{q_i^t (V(1 + A_i^t) - z_i^t)\}$. *The optimal policy minimizing* (15) *performs as:*

(i) *If the inequality* $z_u^t < q_m^t\,(V(1+A_m^t) - z_m^t) - q_l^t\,(V - z_l^t)$ *holds, then it sets* $\mu_l^t = 1$. *Otherwise, it sets* $\mu_m^t = 1$.
(ii) *If* $\mu_l^t = 1$, *then it sets* $\alpha_l^t = 1$.
(iii) *If* $\mu_m^t = 1$, *then it sets* $\alpha_m^t = 0$.

Proof. Assuming that $\mu_i^t = 1$ for a certain $i \in \mathcal{C}_t$, we compare the cost function $\psi(t)$ in two scenarios: one where $\alpha_i^t = 1$ and another

Algorithm 1 Decision-Making of Inner Agent at epoch k

Input: $\{(q_i^{t_k}, A_i^{t_k}, \mathcal{C}_{t_k}, j), r_{k-1}\}$, and $t = t_k$.
1: Send $s_k = \{(q_i^t, A_i^t, \mathcal{C}_t, j), r_{k-1}\}$ to the MBS.
2: Receive content j and γ_j^t in return (run Algorithm 2).
3: Set $\mathcal{C}_{t+1}$ according to (4).
4: Set $\mu_j^t = \alpha_j^t = 1$, $r_k = 0$, and $u = j$.
5: **while** TRUE **do**
6: Update A_u^t, z_u^t, and z_u^t using (1), (9), and (10).
7: Set $r_k = r_k + q_u^t(A_u^t(1-\alpha_u^t)+1)$.
8: Set $t = t+1$.
9: Derive u for which $\mu_u^t = 1$, and α_u^t using Proposition 1.
10: **if** $u \in \bar{\mathcal{C}}_t$ **then**
11: Set $t_{k+1} = t$ and run Algorithm 1 for epoch $k+1$.

where $\alpha_j^t = 1$ for some $j \neq i$, which correspond to $q_i^t\left(V - z_i^t\right) + z_u^t$ and $q_i^t\left(V(1+A_i^t) - z_i^t\right) + z_u^t$, respectively. This comparison reveals that if one content is going to be updated, then content i must be updated (i.e., $\alpha_i^t = 1$). Therefore, given $\mu_i^t = 1$, we must decide between $\alpha_i^t = 0$ and $\alpha_i^t = 1$. Comparing the corresponding cost functions in these two cases, which are $q_i^t\,(V - z_i^t) + z_u^t$ and $q_i^t\,(V(A_i^t+1) - z_i^t)$, respectively, indicates that content i is updated before being multicast (i.e., $\alpha_i^t = 1$) only if it satisfies $z_u^t < V q_i^t\, A_i^t$. Hence, the contents in sets $\mathcal{S}_u^t$ and $\bar{\mathcal{C}}_t$ receive fresh updates when served, while those in $\bar{\mathcal{S}}_u^t$ use the cached copies.

Thus far, we have determined α_i^t given that $\mu_i^t = 1$. Now, we need to decide on $\{\mu_i^t\}$. We initially select the candidate multicast contents l and m from the sets $\mathcal{S}_u^t \cup \bar{\mathcal{C}}_t$ and $\bar{\mathcal{S}}_u^t$, respectively. For any content i in the former set, where $\alpha_i^t = 1$, the associated cost is $\psi(t)|_{1i} = q_i^t\left(V - z_i^t\right) + z_u^t$. Consequently, $l = \arg\min_{i \in \mathcal{S}_u^t \cup \bar{\mathcal{C}}_t} \psi(t)|_{1i}$, as indicated in Proposition 1. In a similar manner, the cost associated with any content i in the second set is $\psi(t)|_{2i} = q_i^t\left(V(A_i^t+1) - z_i^t\right)$. Thus, $m = \arg\min_{i \in \bar{\mathcal{S}}_u^t} \psi(t)|_{2i}$ as per Proposition 1. Finally, by comparing the costs $\psi(t)|_{1l}$ and $\psi(t)|_{2m}$, which leads to the inequality outlined in item (i) of Proposition 1, we identify the content to be multicast. If $\mu_l^t = 1$, then $\alpha_l^t = 1$. Alternatively, if $\mu_m^t = 1$, then $\alpha_m^t = 0$. ■

IV. Outer Agent: Deep Reinforcement Learning

The outer agent deployed in the MBS determines the cache replacement policy, i.e., $\{\gamma_i^t\}$, upon receiving feedback from the SBS (Algorithm 2). More precisely, let us define each epoch as the time between two requests to the MBS for uncached content files. When uncached content is requested, the SBS also sends information to the MBS that includes $s_k = \{\{q_i^{t_k}\}, \mathcal{A}_{t_k}, \mathcal{C}_{t_k}, j_{t_k}\}$ and $r_{t_{k-1}}$, where t_k denotes the index of the first time slot of the k-th epoch, j_{t_k} is the index of the content to be downloaded at t_k, and $r_{t_{k-1}}$ is the reward received at epoch $k-1$. Using this information, the MBS determines the value of $\gamma_j^{t_k} \in \mathcal{C}_{t_k} \cup \{0\}$ that minimizes the expectation $\mathbb{E}[\sum_{k=0}^{\infty} \beta^{t_k} r_{t_k} | s_k]$, where β is the discount factor. The SBS then updates the cache content according to (4), as soon as it receives $\gamma_j^{t_k}$ from the MBS. In what follows, we first cast the content replacement problem at the MBS into an MDP and then propose our DRL-based algorithm to minimize the weighted AAoI derived in (6).

Algorithm 2 Decision-Making of Outer Agent at epoch k

Initialization at $k = 0$**:** $\mathcal{B}$, probability ϵ_0, $\hat{\theta} = \theta = \theta_0$.
1: Receive s_k and r_{k-1} from the SBS.
2: Select $a_{k+1} \in \mathcal{C}_{t_{k+1}} \cup \{0\}$ uniformly at random with probability ϵ_k. Else, $a_{k+1} = \arg\max_a \mathrm{Q}(s_{k+1}, a; \theta_k)$.
3: Send action a_{k+1} to the inner agent.
4: **while** θ_k has not converged **do**
5: Store the tuple (s_k, a_k, r_k, s_{k+1}) in $\mathcal{B}$.
6: **for** every i-th sample in a random mini-batch from $\mathcal{B}$ **do**
7: Set $y_i = r_i + \beta \max_a \mathrm{Q}(s_{i+1}, a; \hat{\theta})$.
8: Set $\theta_k = \theta_k + \zeta\left[y_i - \mathrm{Q}(s_i, a_i; \theta_k)\right] \delta_\theta\, \mathrm{Q}(s_i, a_i; \theta)$.
9: Update $\hat{\theta} = \theta_k$ if $\lfloor \frac{k}{\mathcal{K}} \rfloor = 0$.

A. MDP Problem Formulation

The MDP is represented by a 4-tuple $(\mathcal{S}, \Gamma, \mathcal{P}, \mathcal{R})$, where each element is defined as follows:

- *State space* ($\mathcal{S}$): Set of all possible states, i.e., $\{s_k\}$, where s_k denotes the state at epoch k.
- *Action space* (Γ): Set of actions that can be taken by the MBS at all epochs, where that actions at epoch k include $\gamma_j^{t_k} \in \mathcal{C}_{t_k} \cup \{0\}$.
- *Transition probability matrix* ($\mathcal{P}$): Given the current state s_k at which action $\gamma_j^{t_k}$ is taken, the matrix $\mathcal{P}$ is computed as $\mathcal{P}(s_k, s_{k+1}) = \Pr\left(s_{k+1} | s_k, \gamma_j^{t_k}\right)$.
- *Reward function* ($\mathcal{R}$): The reward received at epoch k is computed as $r_{t_k} = \frac{1}{t_{k+1} - t_k} \sum_{t=t_k}^{t_{k+1}} \sum_{i=1}^{N} q_i^t\, {A'}_i^t\, \mu_i^t$, and is defined as the AAoI per time slot.

A policy π is a (probabilistic) mapping from $\mathcal{S}$ to Γ. The proposed MDP aims to find the optimal π^* such that $\pi^* = \arg\min_\pi \mathbb{E}_\pi[\sum_{k=0}^{\infty} \beta^{t_k} r_{t_k} | s_0]$, where $\mathbb{E}_\pi$ indicates the expectation over policy π. The state and action spaces are yet quite large in practical settings, even if we need to establish maximum values for the AoIs to have a bounded state space. On the other hand, the transition probabilities are determined by the environment, which is, in fact, the inner agent, and are unavailable to the MBS. Thus, the conventional value and policy iteration algorithms are incapable of solving the suggested MDP. As a result, we adopt DRL to overcome the aforementioned challenges.

B. DRL-Based Approach

The state-action value function below represents the total expected discounted reward for selecting action a in state s:

$$\mathrm{Q}_\pi(s, a) = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \beta^{t_k} r_{t_k} | s_0 = s, \gamma_j^{t_0} = a\right]. \tag{16}$$

In Q-learning, the estimate of optimal Q-value function at epoch k is updated as follows, where ζ denotes the learning rate:

$$\mathrm{Q}^{(k+1)}(s_k, a_k) = \mathrm{Q}^{(k)}(s_k, a_k) + \zeta\Big[r_k + \beta \max_a \mathrm{Q}^{(k)}(s_{k+1}, a) - \mathrm{Q}^{(k)}(s_k, a_k)\Big], \tag{17}$$

The action at epoch k is taken by either exploiting the current information in $\mathrm{Q}^{(k)}(s_k, a_k)$, i.e., $a_k = \arg\min_a \mathrm{Q}^{(k)}(s_k, a)$, or

TABLE II
SIMULATION PARAMETERS

Parameter	Value	Parameter	Value
(N, M)	$(30, 5)$	V	0.5
λ	10	$\lvert\mathcal{B}\rvert$	10^4
D in (7)	0.3	Batch size	50
d_i, $\forall i$ in (8)	15	Discount factor (β)	0.95
ϵ decrease rate	0.001	Learning rate (ζ)	0.001

by exploring the space to choose a random feasible action. In DRL, a deep neural network (DNN) with weight θ, denoted by $\mathrm{Q}(s, a; \theta)$, is adopted to approximate the function $\mathrm{Q}(s, a)$, and to avoid computing (17) for $\forall(s, a)$. Consequently, $\mathrm{Q}^{(k)}(s_k, a_k)$ in (17) is approximated with $\mathrm{Q}(s_k, a_k; \theta_k)$, where θ_k is the weight of the DNN at epoch k. The weight θ_k is updated at epoch k through the minimization of the following loss function:

$$L(\theta_k) = \big(y_k - \mathrm{Q}(s_k, a_k; \theta_k)\big)^2, \tag{18}$$

where $y_k = r_k + \beta \max_a \mathrm{Q}(s_{k+1}, a; \hat{\theta})$ is evaluated by the target network $\mathrm{Q}(s, a; \hat{\theta})$. The weights of the target network are set to be equal to the weights of the DNN $\mathrm{Q}(s, a, ; \theta)$ after every $\mathcal{K}$ number of epochs. Accordingly, the update formula for weights θ_k is given as follows, where δ_θ is the gradient with respect to θ:

$$\theta_k = \theta_{k-1} + \zeta \left[y_k - \mathrm{Q}(s_k, a_k; \theta_k)\right] \delta_\theta \, \mathrm{Q}(s_k, a_k; \theta), \tag{19}$$

In Algorithm 2, the outer agent adopts ϵ_k-greedy strategy to choose an action either randomly or from previous experiences (line 3). Moreover, ϵ is configured to decrease with the number of epochs so that the MBS may select the optimal action when the Q-value function is estimated to converge. In addition, an experience replay buffer, $\mathcal{B}$, is used to store the experiences (s_k, a_k, r_k, s_{k+1}). The outer agent then samples a mini-batch of the experiences from memory $\mathcal{B}$ uniformly at random to train the neural network (lines 7 and 8).

V. PERFORMANCE EVALUATION AND DISCUSSIONS

To assess the performance of the proposed optimization approach, we set the simulation parameters as given in Table II. We employed the model in [18] to train the DNN structure in the outer agent, which includes a single fully-connected hidden layer (of size 60). We trained the network over $50,000$ epochs for a batch size of 50, with updates to the target network occurring every 50 epochs. At each time slot, $\lambda = 10$ user requests arrive at the SBS, where each request is associated with content i with the popularity probability p_i. The p_i values are assumed to follow the Zipf distribution with parameter $\alpha_z = 1.5$. To demonstrate the effectiveness of the DRL-based method, we compared it with the MA, LRU, and LFU baseline policies [19].

Fig. 2 compares the AAoI performance per user request (i.e., $\bar{A}_w/\lambda$) of the proposed DRL-based algorithm with policies that exploit three baselines as their outer agent. Note that in MA, the downloaded content replaces the least updated content (i.e., the content with the highest age). We observe in Fig. 2 that the DRL-based method outperforms the LRU, LFU, and MA policies by

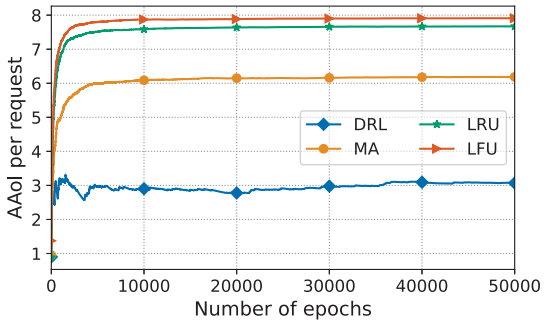
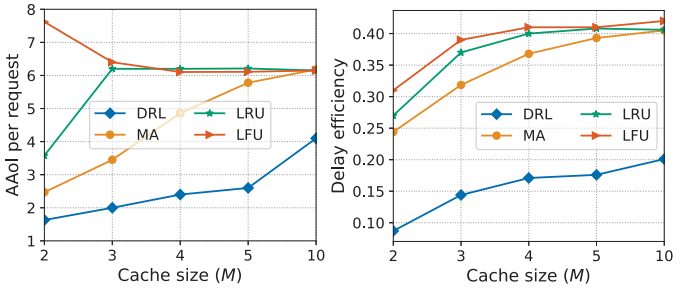


Fig. 2. AAoI per user request versus the number of epochs.

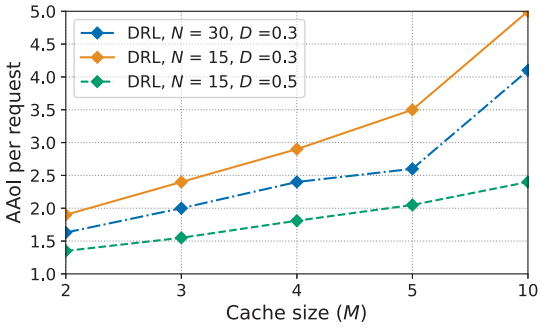


(a) AAoI versus M. (b) Delay efficiency versus M.

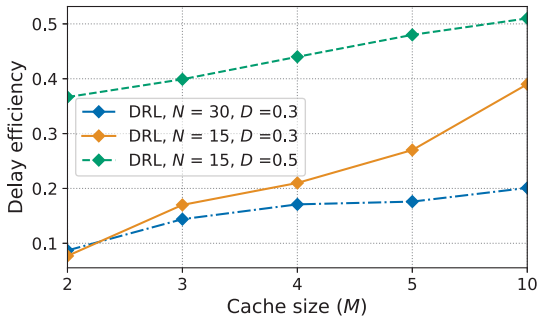
Fig. 3. Performance comparison of (a) AAoI per user request and (b) delay efficiency for varying SBS cache sizes. Here, $N = 30$.

approximately 59%, 60%, and 50%, respectively. This can be attributed to the fact that the baseline policies do not account for the state of the queues. Also, similar to myopic policies, the baselines ignore the impact of their decisions on future epochs, resulting in lower AAoI performance. Furthermore, it is noteworthy that the LFU and LRU policies perform worse than the DRL and MA policies since they discard the least frequently and recently used files regardless of their ages.

Fig. 3 depicts the variation in AAoI per request and delay efficiency against the SBS cache size (M), where the delay efficiency is defined as $\sum_i p_i(d_i - \bar{q}_i)/d_i$, with $\bar{q}_i$ being the average length of queue Q_i. It is observed in Fig. 3a that AAoI increases with M in all policies except LFU, where delay efficiency improves. The latter implies that the users wait less to receive their content. To explain these observations, note that when the cache size is small, most contents need to be downloaded directly from MBS. However, frequent downloads violate the constraints set by the update frequency, as in (7). As a result, the inner agent keeps the requests in their queues for longer periods, leading to lower delay efficiency. On the other hand, as M increases, the inner agent can serve the queues more frequently, albeit with stale cached contents. This results in a higher AAoI but shorter waiting times for the requests. Additionally, the rise in AAoI with increasing M is notably smaller compared to the other baseline policies, while it attains improved delay efficiency across different M values. As M grows, the proportion of updates increases relative to downloads. In LFU, the popularity of cached contents leads to either lower or stable AAoI. Eventually, at larger



(a) AAoI versus M.



(b) Delay efficiency versus M.

Fig. 4. Impact of varying N and D values on (a) AAoI per user request and (b) delay efficiency for varying SBS cache sizes in the DRL-based method.

M values, the performances of all policies converge as cache replacement becomes less impactful.

Fig. 4 shows the impact of key parameters, namely the number of content files (N) and the upper bound of the update rate (D) defined in (7), on the AAoI and delay efficiency in the proposed DRL-based method. In Fig. 4a, it is evident that reducing the value of N from 30 to 15 results in a considerable increase in AAoI at various values of M, thereby highlighting the trade-off between caching and content freshness. Furthermore, increasing D leads to a decrease in AAoI because more fresh content can be downloaded from the MBS. Moreover, as depicted in Fig. 4b, the delay efficiency increases as N decreases to 15 since more popular files can be cached, allowing pending requests related to these contents to be served faster. This increment becomes even more pronounced for larger values of M. Finally, for a fixed N value, increasing D from 0.3 to 0.5 nearly doubles the delay efficiency. This clearly confirms the favorable impact of enhanced update rates on the system performance in the DRL-based strategy.

VI. Conclusion

In this paper, we addressed the joint problem of age-optimal content updating and multicast scheduling in an edge caching system constrained by limited cache capacity, MBS access rate, and average queuing delays of user requests. Our approach to the problem is structured as a two-stage optimization process, where an inner agent utilizes the Lyapunov drift-plus-penalty method to optimize content updating and scheduling decisions at the SBS, while an outer agent employs DRL for cache replacement at the MBS. Simulation results affirm the superiority of our DRL-based cache replacement policy over conventional caching strategies, showcasing improved AAoI per request and delay efficiency. Future research avenues may involve extending the optimization framework to accommodate multiple SBSs and real-world deployment considerations.

References

[1] J. Yao, T. Han, and N. Ansari, "On mobile edge caching," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 3, pp. 2525–2553, Mar. 2019.

[2] S. B. Hassanpour, A. Khonsari, M. Moradian, and S. P. Shariatpanahi, "Privacy-preserving edge caching: A probabilistic approach," *Computer Networks*, vol. 226, p. 109654, 2023.

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

[4] R. D. Yates, P. Ciblat, A. Yener, and M. Wigger, "Age-optimal constrained cache updating," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2017, pp. 141–145.

[5] H. Tang, P. Ciblat, J. Wang, M. Wigger, and R. Yates, "Age of information aware cache updating with file- and age-dependent update durations," in *Proc. IEEE Int. Symp. Model. Opt. in Mobile, Ad Hoc, and Wirel. Netw. (WiOPT)*, Jun. 2020, pp. 1–6.

[6] M. Bastopcu and S. Ulukus, "Information freshness in cache updating systems," *IEEE Trans. Wirel. Commun.*, vol. 20, no. 3, pp. 1861–1874, Mar. 2021.

[7] S. Park, S. Jung, M. Choi, and J. Kim, "AoI-aware Markov decision policies for caching," in *Proc. IEEE Int. Conf. Distr. Comput. Syst. (ICDCS)*, Jul. 2022, pp. 1274–1275.

[8] B. Abolhassani, J. Tadrous, A. Eryilmaz, and E. Yeh, "Fresh caching of dynamic content over the wireless edge," *IEEE/ACM Trans. Netw.*, vol. 30, no. 5, pp. 2315–2327, Oct. 2022.

[9] G. Ahani and D. Yuan, "Accounting for information freshness in scheduling of content caching," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2020, pp. 1–6.

[10] M. Ma and V. W. Wong, "Age of information driven cache content update scheduling for dynamic contents in heterogeneous networks," *IEEE Trans. Wirel. Commun.*, vol. 19, no. 12, pp. 8427–8441, Dec. 2020.

[11] S. Zhang, L. Wang, H. Luo, X. Ma, and S. Zhou, "AoI-delay tradeoff in mobile edge caching with freshness-aware content refreshing," *IEEE Trans. Wirel. Commun.*, vol. 20, no. 8, pp. 5329–5342, Aug. 2021.

[12] N. Chukhno, O. Chukhno, D. Moltchanov, A. Gaydamaka, A. Samuylov, A. Molinaro, Y. Koucheryavy, A. Iera, and G. Araniti, "The use of machine learning techniques for optimal multicasting in 5G NR systems," *IEEE Trans. Broadcast.*, vol. 69, no. 1, pp. 201–214, Mar. 2023.

[13] H. Seo, K. Son, S. Park, and W. Choi, "Communication-efficient private information acquisition: Multicasting via crowding," *IEEE Trans. Veh. Technol.*, vol. 70, no. 7, pp. 7199–7204, Jul. 2021.

[14] 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.

[15] H. Hao, C. Xu, M. Wang, L. Zhong, and D. O. Wu, "Stochastic cooperative multicast scheduling for cache-enabled and green 5G networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2019, pp. 1–6.

[16] L. Zhong, C. Xu, J. Chen, W. Yan, S. Yang, and G.-M. Muntean, "Joint optimal multicast scheduling and caching for improved performance and energy saving in wireless heterogeneous networks," *IEEE Trans. Broadcast.*, vol. 67, no. 1, pp. 119–130, Mar. 2021.

[17] L. Georgiadis, M. J. Neely, and L. Tassiulas, "Resource allocation and cross-layer control in wireless networks," *Found. Trends Netw.*, vol. 1, no. 1, pp. 1–144, Apr. 2006.

[18] M. A. Abd-Elmagid, A. Ferdowsi, H. S. Dhillon, and W. Saad, "Deep reinforcement learning for minimizing age-of-information in UAV-assisted networks," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2019, pp. 1–6.

[19] J. Zhu, R. Li, G. Ding, C. Wang, J. Wu, Z. Zhao, and H. Zhang, "AoI-based temporal attention graph neural network for popularity prediction and content caching," *IEEE Trans. Cogn. Commun. Netw.*, vol. 9, no. 2, pp. 345–358, Apr. 2023.