Caching in Dynamic Environments: A Near-Optimal Online Learning Approach

Shiji Zhou, Zhi Wang, *Member, IEEE*, Chenghao Hu, Yinan Mao, Haopeng Yan, Shanghang Zhang, *Member, IEEE*, Chuan Wu, *Senior Member, IEEE*, and Wenwu Zhu, *Fellow, IEEE*

***Abstract*—The rapid growth of rich multimedia data in today's Internet, especially video traffic, has challenged the content delivery networks (CDNs). Caching serves as an important means to reduce user access latency so as to enable faster content downloads. Motivated by the dynamic nature of the real-world edge traces, this paper introduces a *provably well* online caching policy in dynamic environments where: 1) the popularity is highly dynamic; 2) no regular stochastic pattern can model this dynamic evaluation process. First, we design an online optimization framework, which aims to minimize the *dynamic regret* that finds the distance between an online caching policy and the best dynamic policy in hindsight. Second, we propose a dynamic online learning method to solve the non-stationary caching problem formulated in the previous framework. Compared to the linear dynamic regret of previous methods, our proposal is proved to achieve a *sublinear dynamic regret*, from which it is guaranteed to be nearly optimal. We verify the design using both synthetic and real-world traces: the proposed policy achieves the best performance in the synthetic traces with different levels of dynamicity, which verifies the dynamic adaptation; our proposal consistently achieves at least 9.4% improvement than the baselines, including LRU, LFU, Static Online Learning based replacement, and Deep Reinforcement Learning based replacement, in random edge areas from real-world traces (from iQIYI), further verifying the effectiveness and robustness on the edge.**

***Index Terms*—Dynamic environment, dynamic regret, online learning.**

Manuscript received 31 March 2021; revised 23 July 2021, 17 September 2021, and 26 October 2021; accepted 15 November 2021. Date of publication 2 December 2021; date of current version 9 March 2023. This work was supported in part by the National Key Research and Development Program of China under Grant 2020AAA0106300, in part by the National Natural Science Foundation of China under Grants 62050110 and 61872215, in part by the Shenzhen Science and Technology Program under Grant RCYX20200714114523079, and in part by the Kuaishou for sponsoring the research. The Associate Editor coordinating the review of this manuscript and approving it for publication was Dr. Ali C. Begen. *(Corresponding authors: Zhi Wang and Wenwu Zhu.)*

Shiji Zhou is with the Tsinghua-Berkeley Shenzhen Institute, Shenzhen 518000, China (e-mail: zsj17@mails.tsinghua.edu.cn).

Zhi Wang is with the Tsinghua Shenzhen International Graduate School and Peng Cheng Laboratory, Shenzhen 518000, China (e-mail: wangzhi@sz.tsinghua.edu.cn).

Chenghao Hu is with the Electrical & Computer Engineering, University of Toronto, Toronto, Ontario M4Y1M7, Canada (e-mail: ch.hu@mail.utoronto.ca).

Yinan Mao is with the Tsinghua Shenzhen International Graduate School, Tsinghua University, Shenzhen 518000, China (e-mail: 525067649@qq.com).

Haopeng Yan is with the Department of Computer Science and Technology, Tsinghua University, Beijing 100080, China (e-mail: yhjp18@mails.tsinghua.edu.cn).

Shanghang Zhang is with EECS, UC Berkeley, Berkeley, CA USA (e-mail: shz@eecs.berkeley.edu).

Chuan Wu is with the Department of Computer Science, University of Hong Kong, Hong Kong (e-mail: cwu@cs.hku.hk).

Wenwu Zhu is with the Department of Computer Science and Technology, Tsinghua University, Beijing 100080, China (e-mail: wwzhu@tsinghua.edu.cn).

Digital Object Identifier 10.1109/TMM.2021.3132156

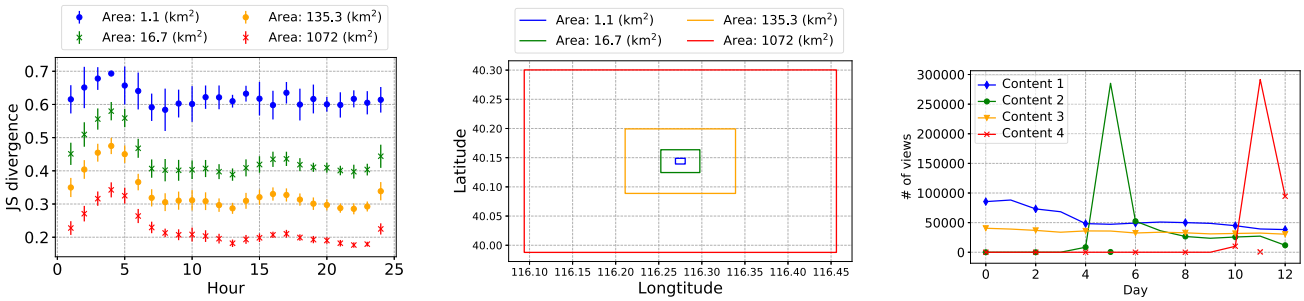
I. Introduction

THE INTRODUCTION is divided into three parts to elaborate on the general context of this section. We will start with the background and motivation, then analyze some of the most relevant works, and finally summarize our method and main contributions.

A. Background and Motivation

The explosion of multimedia applications has caused significant backbone traffic in content distribution, bringing the demand for high-quality multimedia content to an unprecedented level, *e.g.* video traffic is predicted to reach 77.5 EB/month by 2022 [13]. This challenges the traditional centralized content distribution architectures. Using edge content distribution has thus been proposed to improve service capacity and reduce backbone bandwidth consumption [8], as traffic is *localized* since the content is cached and served from edge devices (*e.g.* base stations, home wireless routers) closed to the users.

As compared to conventional centralized content delivery networks with dedicated data centers for a large region (*e.g.* a city), edge content devices usually have much more limited service range (*e.g.* a college or even a building). Previous measurement results show that requests can be served within several hundred meters if cellular base stations can be utilized as edge content caches [10]. As a result, such edge content serving makes requests extremely *sparse* at each edge device, and the request patterns include both the *popularity distribution* and *request volume* vary tremendously over time. Using trace-driven measurement, we show the *non-stationary* nature of requests captured in a real-world trace – the iQIYI dataset [8]. In Fig. 1(a), we use the JS divergence to measure the difference between popularity distributions of two sequential hours in each day, with different area sizes varying from 1072 km^2 to 1.1 km^2 as described in Fig. 1(b). The JS divergence, which is a measure of the similarity of different distributions, between two sequential hours is over 0.2, suggesting that the cached contents at an edge area are changing continuously over time. When the size of an edge area is smaller (i.e. "edgeness" increases), the JS divergence becomes larger, suggesting that edge cache encounters more dynamical request change. In Fig. 1(c), the curves represent the number of requests of selected contents over time. We observe that the



(a) JS divergence (b) Geographic coordinates (c) Number of requests to contents

Fig. 1. (a) JS-divergence between popularity distributions of two sequential hours in a day for different edge area sizes. (b) Geographic coordinates of the corresponding areas in (a). (c) Number of requests to four different videos in the same edge area in two weeks.

TABLE I
COMPARISON WITH MOST RELATED WORK

Solution	No prior assumption	Stationary guarantee	Non-stationary guarantee
Model-based	✗	✗	✗
Deep RL	✓	✗	✗
Static OL	✓	✓	✗
Ours	✓	✓	✓

"Deep RL" represents deep reinforcement learning based policy and "Static OL" represents static online learning. There is no previous method that guarantees the performance in both stationary environments where the distribution pattern is unknown and non-stationary without prior assumption.

different contents have dramatically different request patterns, suggesting that no uniform distribution pattern can model all the contents. The results suggest that the real-world requests are *highly dynamic without regular stochastic pattern especially on the edge*, which motivates us to seek a solution that *robustly performs well in such non-stationary scenarios*.

B. Most Related Work

In contrast with the stationary environment where the popularity remains unchanged, the non-stationary property has challenged previous dynamic content replacement solutions, as summarized in Table I.

Model-based methods, including offline machine learning methods [17]–[19] and conventional schemes, such as Least Frequently Used (LFU), Least Recently Used (LRU), and their variants [6], are based on assumptions of *static or regular popularity distribution*, as they usually assume stationary [11], regular evaluation (*e.g.* Poisson Noise Model [14], Independent Reference Model [5]), or adversary request [12] pattern. They are claimed to be optimal by maximizing the hit rate expectation under the above distribution assumptions, which are not true in edge content delivery. Thus, their optimization results may be invalid in real-world traces, where the popularity of content has no regular patterns as they assumed.

Deep Reinforcement Learning (DRL) methods [20]–[24], [26], [27], [37], which address the dynamic challenge with empirical designs (*e.g.* neural networks, hyperparameters), are claimed to perform well in dynamic settings without prior knowledge. However, since Deep RL-based methods have no strict theoretical guarantee and rely on hyperparameters, DRL may only work well in particular situations by relying on dedicated models.

Static Online Learning (OL) methods [9], [40] attempt to minimize the *static regret*, which is the performance gap between an online learning algorithm and the best static policy in hindsight. Static OL does not make any assumptions on the popularity pattern, and has a provably guarantee to fare against the *best static policy*, thus is claimed to be nearly optimal. However, since the popularity distribution varies rapidly over time (especially on the edge) as we studied, the best static policy that makes no content replacement will not perform well in the real-world traces. Therefore, faring against the *best static policy* would not guarantee the non-stationary performance.

To sum up, our method could give out a provable guarantee of the performance, do well in adapting to the dynamic edge environment and do not make any prior assumptions to generalize to different content patterns, compared to other previous methods.

C. Methodology and Contributions

In this paper, we aim to design a *provably well caching algorithm for dynamic environments with no prior distribution assumptions*. To achieve this, we first design an online optimization framework for measuring the caching performance based on the *dynamic regret* which focuses on dynamic scenarios and better to build up the model [44], which is the distance between an online learning algorithm and the *best dynamic policy* in hindsight. We prove that a caching policy with *sublinear dynamic regret* is guaranteed to approach the *best dynamic policy* in hindsight; otherwise, those with linear dynamic regret would suffer from performance decay in some cases. Then we propose an *adaptive* Nearly Optimal Cache (NOC) replacement strategy based on dynamic online learning, aiming to achieve *sublinear dynamic regret*. NOC runs an Online Gradient Descent (OGD) algorithm with dynamic stepsize that comes from our novel analysis, where we give a general dynamic regret bound for OGD-liked methods.

In our theoretical results, we prove NOC to have a sublinear dynamic regret, from which our solution is promising to adaptively handle and perform well in different request patterns, including static and dynamic circumstances. Besides, previous

methods, including conventional policies and static online learning policy [40], are proved to have linear dynamic regret, and thus work badly in dynamic environments. Experimentally, we first verify our design's dynamic adaptation by synthetic traces with different levels of dynamicity. NOC performs well in static traces and achieves the best performance in different dynamic situations, which supports the claims in our analysis. We then verify the edge effectiveness and robustness in real-world traces (from iQIYI) with different area sizes and random locations. NOC consistently outperforms in all area sizes, and gets more than 9.4% (averaging over randomly sampled locations) improvement than other baselines, including LRU, LFU, Static Online Learning based policy – OGA [40] and Deep Reinforcement Learning based policy – DRL [20].

As a study that is the most similar to our proposal, OGA [40] uses an OGD-based method[1] to minimize the *static regret*, which ignores the dynamic goal and loses generality in dynamic environments. From this point of view, our fundamental idea of algorithm design and analysis has nothing in common with the previous one. To the best of our knowledge, there is no previous solution that targets the dynamical request patterns in edge content distribution, with a strict theoretical dynamic regret guarantee. The contributions of our study are summarized below.

▷ We propose a new online optimization framework for online caching based on the dynamic regret, which practically captures the distance between a cache replacement strategy and the best dynamic policy in hindsight. In this framework, an online caching strategy with a sublinear dynamic regret is guaranteed to have nearly optimal performance with a provable theoretical guarantee.

▷ We design an adaptive dynamic stepsize for online gradient descent that minimizes the dynamic regret accordingly, and propose an adaptive Nearly Optimal Cache (NOC) caching strategy. In our analysis, NOC is proved to achieve a sublinear dynamic regret of $O(T^{(1+\delta)/2}), \delta \in [0,1)$, which guarantees NOC to perform well in different patterns to overcome the edge complex situation. In contrast with conventional strategies such as LRU, LFU, LRU-k [1], LRFU [7], and sublinear static regret strategies [9], [40], we give out their linear dynamic regret lower bound, from which their performances are consistently worse than NOC in dynamic environments in the worst case.

▷ We use synthetic traces and real-world traces (from iQIYI) to verify our design: NOC works well for synthetic traces with both the static pattern and different degrees of dynamicity, which verifies the dynamic adaptation and supports our theoretical analysis. We then evaluate our solution on iQIYI traces with different area sizes and locations. NOC keeps obvious outperformance than baselines, including LRU, LFU, OGA, and DRL, which verifies the effectiveness and robustness on edge.

The rest of the paper is organized as follows. We provide the system model and online optimization framework in Section II. We propose the algorithm, analysis and theoretical comparisons in Section III. We evaluate our design in Section IV. We survey related work in Section V and conclude the paper in Section VI.

[1] In [40], the online algorithm ascends a concave function, which is equivalent to descend its negative counterpart.

II. Problem Formulation

In this section, we formulate the edge cache replacement problem with dynamical request patterns as an online learning problem.

A. Non-Stationary Online Learning Framework

At each time step $t = 1, \dots T$, the online learner takes a decision y_t in a convex set $\mathcal{K}$. After, the environment reveals a convex loss function $l_t : \mathcal{K} \to \mathbb{R}$, and the online learner suffers a loss $l_t(y_t)$. The dynamic regret is a theoretical metric for the performance of online algorithms in changing environments, defined as

$$\text{DA-Regret}_T = \sum_{t=1}^{T} l_t(y_t) - \sum_{t=1}^{T} l_t(u_t),$$

where $\{u_t, t = 1, \dots, T\}$ is the dynamic competitor in hindsight. Achieving a sublinear dynamic regret bound is not possible unless specific constraints are made about the competitor [44]. A famous one is the *path-length* [44]:

$$V(T) = \sum_{t=1}^{T-1} \|u_t - u_{t+1}\|,$$

which measures the degree of freedom of the competitor. Usually, we assume $V(T) \leq O(T^{\delta}), \delta \in [0,1)$ since when the $V(T) \geq O(T)$, it is impossible to achieve a sublinear dynamic regret [44].

B. System Model

In the edge cache system, we assume that there is a content set $\mathcal{N} = \{1, 2, \dots, N\}$, to be requested by users. Without loss of generality, we assume these contents are of the same size 1, and each content item can be cached by the edge device with capacity C. We use $\boldsymbol{x}_t, t \in \{1, 2, \dots, T\}$ to denote the sequential requests received at an edge cache device, where t is the index of the request in the sequence; And $\boldsymbol{x}_t = (\boldsymbol{x}_t^1, \boldsymbol{x}_t^2, \dots, \boldsymbol{x}_t^N)$, is a one-hot vector to identify the content item requested by the t-th request, where $\boldsymbol{x}_t^{n_t} = 1$ if n_t is the only content item being requested at time slot t and $\boldsymbol{x}_t^i = 0, i \neq n_t$. We further use a cache status vector $\boldsymbol{y}_t = \{\boldsymbol{y}_t^1, \boldsymbol{y}_t^2, \dots, \boldsymbol{y}_t^N\}$ for the ease of presentation, where $\boldsymbol{y}_t^i \in [0,1], \sum_i \boldsymbol{y}_t^i \leq C$ and $\boldsymbol{y}_t^i$ denotes the caching probability of file i.[2]

Therefore, a cache performance function $f_t(\boldsymbol{y}_t)$ can be defined to measure how well the edge cache performance is, which can be hit rate [40], bandwidth economization [2], QoS improvement [3], or any other concave evaluation. Without loss of generality, we use cache hit rate in our study, $f_t(\boldsymbol{y}_t) = \boldsymbol{x}_t^{\intercal} \boldsymbol{y}_t$. We also define a caching policy by σ (e.g. LRU, LFU), and $\boldsymbol{y}_t(\sigma)$ to be the cache status vector provided by caching policy σ by the t-th request.

[2] In practical systems, $\boldsymbol{y}_t^i$ should be rounded, i.e., the policy caches the contents with the highest probability $\boldsymbol{y}_t^i$, while replacing the content with the lowest probability. Similar modeling can be referred to previous work [40].

C. Best Dynamic Policy in Hindsight

In the ideal case, we aim to find the the *best dynamic policy* σ^* with $V(T) \leq O(T^\delta), \delta \in [0,1)$ times of replacements, to maximize the average cache performance over time, as follows.

$$\begin{aligned} \sigma^* &= \arg\max_{\sigma} \frac{1}{T}\sum_{t=1}^{T} f_t(\boldsymbol{y}_t(\sigma)) \\ s.t.\ \ & \boldsymbol{y}_t(\sigma) \in [0,1]^N, \|\boldsymbol{y}_t(\sigma)\|_1 \leq C, t = 1,2,\ldots,T \\ & \sum_{t=1}^{T-1} \|\boldsymbol{y}_{t+1}(\sigma) - \boldsymbol{y}_t(\sigma)\| \leq V(T). \end{aligned}$$

Note that the above formulation is only for a "conceptual" content replacement, and does not work for practical systems, which can be viewed as the competitor in hindsight as non-stationary online learning framework. It makes decisions by using the "future" information in advance, while a practical solution should generate decisions only from the past and current information. Therefore, since the precise best policy is not achievable in practice, we intend to propose an approximated framework that aims to achieve *nearly the best performance* via dynamic online learning.

Remark: Since the best dynamic policy in hindsight is clairvoyant, the constrain of the cache replacement is necessary in the ideal case (not for practice), or the optimization problem will be meaningless as

$$\max_{\sigma} \frac{1}{T}\sum_{t=1}^{T} f_t(\boldsymbol{y}_t(\sigma)) = \frac{1}{T}\sum_{t=1}^{T} \max_{\boldsymbol{y}_t} f_t(\boldsymbol{y}_t),$$

which means the hit rate is 100% for any situations (the clairvoyant would always cache the contents of future requests), and obviously, it is not attainable in practice.

D. Dynamic Regret of Caching Policy

To this end, we refine the above ideal model to the practical approximated model, so that it can practically fit the real-world edge cache scenario.

1) Definition of Dynamic Regret: We introduce the dynamic regret that measures the gap between a practical caching policy σ that makes decisions without "future" knowledge, and the ideal *best dynamic policy* σ^* (as defined previously) in hindsight. More specifically, we define the *dynamic regret* of caching policy σ as:

$$\text{DA-Regret}_T(\sigma) = \sum_{t=1}^{T} f_t(\boldsymbol{y}_t(\sigma^*)) - \sum_{t=1}^{T} f_t(\boldsymbol{y}_t(\sigma)).$$

2) Relationship Between Dynamic Regret and Caching Performance: We present the following theorem to describe the relationship between the proposed dynamic regret and caching performance.

Theorem 1: If a caching policy has a sublinear dynamic regret upper bound:

$$\text{DA-Regret}_T(\sigma) \leq O(T^\alpha), \alpha \in [0,1),$$

with which, we can bound the gap of average performance (e.g. average hit rate) between the optimal policy σ^* and online policy σ by

$$\frac{1}{T}\sum_{t=1}^{T} f_t(\boldsymbol{y}_t(\sigma^*)) - \frac{1}{T}\sum_{t=1}^{T} f_t(\boldsymbol{y}_t(\sigma)) \leq O(1/T^{1-\alpha}).$$

The proof is direct from the definition. By Theorem 1, we can claim that if a caching policy has a sublinear dynamic regret bound, it is proved to achieve an approximated performance with the best policy in hindsight when $T \to \infty$. We thus call it *nearly optimal cache policy.* In addition, as $1-\alpha$ represents the convergence rate to the optimal performance, we want α to be as small as possible.

Theorem 2: If a caching policy has a linear dynamic regret lower bound, i.e.

$$\text{DA-Regret}_T(\sigma) \geq O(T),$$

with which, we have

$$\frac{1}{T}\sum_{t=1}^{T} f_t(\boldsymbol{y}_t(\sigma^*)) - \frac{1}{T}\sum_{t=1}^{T} f_t(\boldsymbol{y}_t(\sigma)) \geq O(1).$$

The proof is direct from the definition. By Theorem 2, an online policy with linear dynamic regret would be a constant worse than the optimal policy σ^*, thus works badly in some cases.

3) Comparison With the Static Regret: The main difference between static regret [40] and dynamic regret is the definition of "competitor policy" that is chosen to compete against. Specifically, the static regret compares the online performance with a "static competitor policy" that makes no cache changes, and the dynamic regret measures the bias with a "dynamic competitor policy" that is able to replace the cached contents. Minimizing the static regret and approximating the best static policy is ineffective when dealing with non-stationary settings. Therefore previously used static regret [40] is not suitable for the dynamic caching problem.

III. Nearly Optimal Solution

In this section, we first introduce a Nearly Optimal Cache policy (Algorithm 1) based on Online Gradient Descent (OGD) [43], and design our dynamic stepsize that fits the dynamic environments. Then, we prove that our solution has a dynamic regret bound sublinear to T and guarantees to approach the optimal policy in hindsight, which outperforms LFU, LRU, LRU-based schemes and static online learning methods as we show that they are not able to approach such optimal policy.

A. Nearly Optimal Cache (NOC)

In our study, we aim to design an algorithm that achieves a *sublinear bound* for all $T = 1,2,\ldots,\infty$, with the following goals: 1) the algorithm can minimize the dynamic regret defined previously with a sublinear bound. 2) the parameter selection should be *adaptive* and does not depend on T.

Algorithm 1: Nearly Optimal Cache (NOC).

Input: Cache capacity C, dynamic controller δ.
Initial: $\boldsymbol{y}_1 = \mathbf{0}$.
for $t = 1, \ldots, T$ **do**
  Receive cache request $\boldsymbol{x}_t$, and reveal the cache hit $f_t(\boldsymbol{y}_t)$.
  Calculate dynamic stepsize $\eta_t = \frac{1}{t^{(1-\delta)/2}}$.
  Update $\boldsymbol{y}_{t+1} \leftarrow \Pi_{\boldsymbol{y}\in\mathcal{K}_C}(\boldsymbol{y}_t + \eta_t \nabla f_t(\boldsymbol{y}_t))$.
  Replace the cached content i with the lowest $\boldsymbol{y}_t^i$ by the noncached content j with highest $\boldsymbol{y}_t^j$, if $\boldsymbol{y}_t^j > \boldsymbol{y}_t^i$.
end for

1) Online Gradient Descent Framework: In our design, we use the Online Gradient Descent (OGD) iteration as follows:

$$\boldsymbol{y}_{t+1} \leftarrow \Pi_{\boldsymbol{y}\in\mathcal{K}_C}\left(\boldsymbol{y}_t + \eta_t \nabla f_t(\boldsymbol{y}_t)\right),$$

where η_t is the dynamic stepsize [43], $\mathcal{K}_C = \{\boldsymbol{y}_t \in [0,1]^N, \|\boldsymbol{y}_t\|_1 \le C\}$ is the valid exploration space for $\boldsymbol{y}_t$ as the cached contents will not exceed the cache capacity, and $\Pi_{\mathcal{K}_C}(\cdot)$ is a projection function that maps $\boldsymbol{y}_t + \eta_t \nabla f_t(\boldsymbol{y}_t)$ to a valid cache decision.

We take the same projection method as [40], where the total computational cost is $O(NT)$ as there is only one request in each round, and the memory cost is $O(N)$. Therefore the computational cost is equivalent to LRU, and the memory cost is the same as LFU, which maintains the same efficiency as conventional methods.

2) Dynamic Stepsize: With the above iteration scheme, the main task is then to design an *adaptive dynamic stepsize* that achieves the goals defined above. We first use a control parameter δ such that $V(T) \le O(T^\delta), \delta \in [0,1)$. Next, we provide our dynamic stepsize as follows.

$$\eta_t = \frac{1}{t^{(1-\delta)/2}}. \tag{1}$$

Though the formulation is simple, it is the key factor to guarantee a novel effect and requirement of the optimal approach. The dynamic step size design is not straightforward and actually comes from a strict dynamic regret analysis. We will show its sublinear dynamic regret guarantee and analyze its performance against conventional solutions as following.

B. Analysis of Dynamic Regret

In this subsection, we show the sublinear dynamic regret bound is achievable with respect to the adaptive dynamic stepsize.

1) General Dynamic Regret Bound: We first analyze the parameterized bound for different choices of the dynamic stepsize η_t in general, which helps the design of a nearly optimal solution.

Theorem 3: (General bound). Assume there exists a fixed constant G such that $\max_t \|\nabla_t f_t(\boldsymbol{y}_t)\| \le G$. For any algorithm using online gradient descent with a non-increasing dynamic stepsize $\eta_t > 0$, it has the following upper bound for all T.

$$\text{DA-Regret}_T \le \frac{C}{\eta_T} + \frac{V(T)}{2\eta_T} + \frac{\sum_{t=1}^T \eta_t}{2} G^2.$$

The proof detail is provided in Appendix A1. To achieve a nearly optimal solution, the dynamic step size must satisfy both $\frac{V(T)}{2\eta_T}$ and $\sum_{t=1}^T \eta_t \le O(T^\alpha), \alpha \in [0,1)$.

2) Dynamic Regret of Our Design: We next verify our design as following.

Theorem 4: (Dynamic Regret of NOC). Assume our control parameter δ satisfies that $V(T) \le O(T^\delta)$. If the adaptive dynamic stepsize follows our design in Eq. 1, where $\eta_t = \frac{1}{t^{(1-\delta)/2}}$, it will achieve a *sublinear* dynamic regret bound *for all* T as follows.

$$\text{DA-Regret}_T(\text{NOC}) \le O(T^{(1+\delta)/2}).$$

The proof detail is provided in Appendix A2. By Theorem 4, with a suitable choice of δ, we can always approach the optimal performance when $T \to \infty$, which implies that our proposed NOC policy has nearly the same average performance (e.g., hit rate) as the optimal policy.

Setting of δ: From Theorem 4, the parameter δ affects the theoretical performance of NOC. Since the $V(T)$ depends on the dynamics of the environment,[3] we could increase the value of δ in a more dynamic environment, and decrease this value in a relatively stable environment. In practice that has no prior knowledge of suitable δ, we could just set $\delta = 1/2$ as default that meets all situations with less than $O(T^{3/4})$ regret when $V(T) \le O(\sqrt{T})$.

Remark: We do not make any distribution assumption on the request patterns, which means this upper bound holds for any kind of request sequence and better fit the edge scenarios' practical need.

C. Comparison With Previous Methods

1) Comparison With Conventional Schemes: Having analyzed the regret upper bound of our proposed design, we next characterize the performance of conventional schemes including LFU and LRU, which can be generalized to LRU-k [1], LRFU (Least Recently Frequently Used) [7].

Note that LFU, which uses cumulative frequency counts, is equivalent to FTRL (Follow the Regularized Leader) with any step size parameters. Thus, it achieves a *Tight Static Regret* via tuning the parameters by Theorem 5.1 of [43], from which LFU can quickly achieve the optimal performance in static environments. However, such a guarantee is invalid in dynamic situations.

Theorem 5 (Dynamic Regret of LFU): The LFU policy has the following dynamic regret lower bound:

$$\text{DA-Regret}_T(\text{LFU}) \ge O(T).$$

The proof detail is provided in Appendix A3.

Theorem 6 (Dynamic Regret of LRU): The LRU policy has the following dynamic regret lower bound

$$\text{DA-Regret}_T(\text{LRU}) \ge O(T).$$

[3]The optimal caching policy σ^* caches the most popular contents. In the dynamic environment, the content popularity is changing, thus the popular contents change as well. Then the optimal caching policy σ^* would replace the out-of-date contents with new popular contents. When this dynamic is severer, the number of replacements $V(T)$ becomes larger.

The proof detail is in Appendix A4. The proof can be extended directly to variations of LRU, such as LRU-k [1], LRFU (Least Recently Frequently Used) [7], which also have linear regret.

From the above analysis and Theorem 2, LFU, LRU, and LRU-based policies have $O(T)$ dynamic regret, which implies they cannot give a considerable performance in non-stationary settings.

2) Comparison With Static Online Learning: We further compare our proposal to static online learning-based solutions. In particular, we study the previous Online Gradient Ascent (OGA) [40] and Follow the Perturbed Leader (FTPL)-based [9] solutions.

The OGA policy takes a fixed stepsize $\eta_t = \frac{\sqrt{2C}}{\sqrt{T}}$ with OGD framework. Though it achieves sublinear static regret, it has linear dynamic regret as following.

Theorem 7 (Dynamic Regret of OGA): The OGA policy that minimizes a static regret has the following lower bound for dynamic regret:

$$\text{DA-Regret}_T(\text{OGA}) \geq O(T).$$

The proof detail is provided in Appendix A5.

The FTPL-based policy uses cumulative frequency counts with added Gaussian noise for caching decisions. The analysis shows that the FTPL policy has near-optimal static regret. However, its expected (it is a stochastic method) dynamic regret is linear as the following theorem.

Theorem 8 (Dynamic Regret of FTPL): The FTPL policy that minimizes a static regret has the following lower bound for dynamic regret:

$$E[\text{DA-Regret}_T(\text{FTPL})] \geq O(T).$$

The proof detail is in Appendix A-F. According to Theorem 7 and Theorem 8, minimizing the static regret can not solve the dynamical content requests in the edge cache situation. However, in the real-world caching scenarios, dynamical request patterns are common as shown in our measurement studies before. Therefore such static online learning-based policies are less efficient in handling the cache replacement.

IV. Performace Evaluation

In this section, we evaluate our proposal using simulation experiments based on synthetic traces and real-world traces from iQIYI. First, we introduce the traces, baselines and parameter settings in the experimental setup. Second, we verify the dynamic adaptation of the proposed method with the synthetic traces. Third, we verify NOC's real-world performance from the perspective of both "edgeness" (how small the edge area is) and "robustness" (how stable the performance is).

A. Experimental Setup

1) Traces: Our simulation experiments are driven by the following traces.

Synthetic Traces: to evaluate our design's performance in corner cases that cannot be well captured by real-world traces, we also generate other request patterns. a) Zipf traces generated by an independent and identically distributed (i.i.d.) Zipf model [28] to model pure stationary environments; b) Poisson Zipf traces generated by a Poisson shot noise model [14] to model continuously changing environments; c) Zipf-Shuffle traces generated by a shuffling Zipf model (varying across time slots) to model suddenly changing environments.

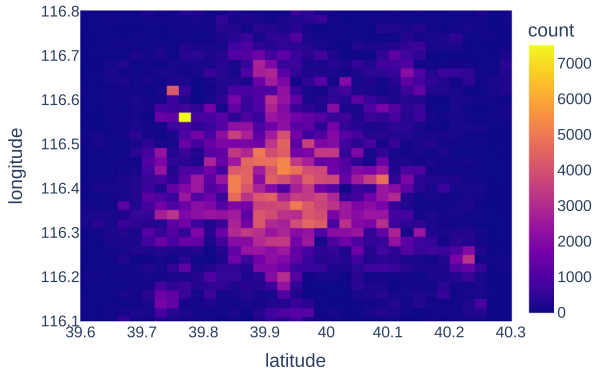


Fig. 2. The geographic distribution of requests from iQIYI traces.

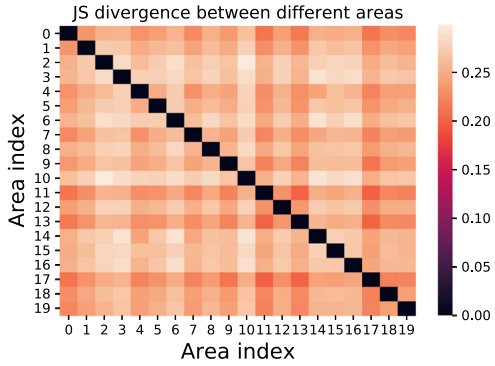


Fig. 3. The JS-divergence between the request distributions of different areas. The x-axis and y-axis represent the area index of the selected areas, and the color depth represents the JS divergence between the request distribution area x and y. The JS-divergence value shows that the content distributions are different from each other.

Traces from iQIYI: we use the iQIYI traces, which record not only the video requests but also the locations of the requests. It contains 53954230 requests from 417077 contents in the whole Beijing during 14 days. We show the request-location distribution in Fig. 2. We can map them to different edge devices: a) We vary the *service range* of an edge cache device from $15.5 \times 17.4\text{km}^2$ to 2.0×2.2 km^2 to evaluate the impact of the "edgeness" on our proposal's performance; b) We randomly select 20 representative areas with 2×2 km^2 area size as the edge regions to verify the "robustness" of NOC, since the request distributions are different among them as shown in Fig. 3. The possible reason is that different communities may have different preferences. For example, the college community prefers different from the CBD (Central Business District) community. We study the single cache problem, and all representative areas only deploy one edge server.

2) Baselines: We compare our design with the following baseline solutions. a) Least Recently Used (LRU): the edge cache device caches the most recently used content in the cache and evicts those that are least recently used. b) Least Frequently Used (LFU): the edge cache device caches the most frequently

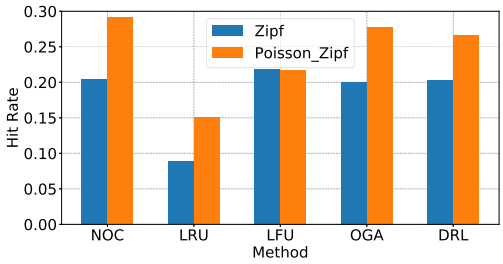
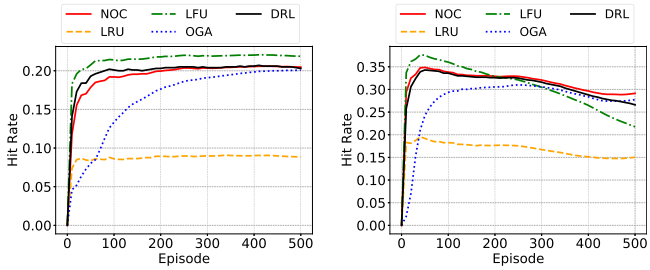
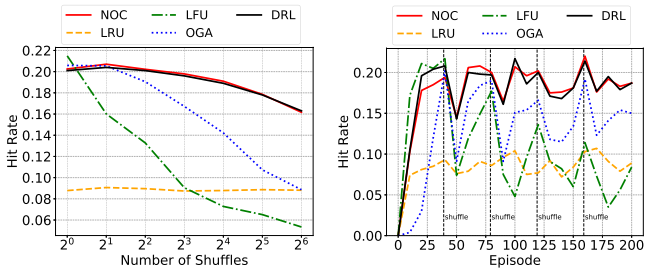


Fig. 4. The average hit rate of the NOC policy and the baseline algorithms.



(a) Zipf (b) Poisson Zipf

Fig. 5. Average cache hit rate versus episode in: (a) Zipf traces. (b) Poisson Zipf traces.



(a) Hit v.s. Changes (b) Hit v.s. Time

Fig. 6. Hit rate evaluation of NOC and other baselines under different shuffle times in: (a) Zipf-Shuffle traces with different shuffle times. (b) One trace with 16 times of shuffles.

used content in the cache and evicts those that are least frequently used. c) Online Gradient Ascent (OGA) [40]: a previous online learning-based policy that aims to minimize the static regret. d) Deep Reinforcement Learning (DRL) [20]: a Deep Reinforcement Learning-based policy with Wolpertinger architecture that aims at maximizing the long-term cache hit rate. We use the same parameter settings as in the paper [20].

Remark: Note that we do not compare with the FTPL policy [9], since FTPL needs to sample N dimensional normal vector in each round, which makes the computational cost too large for evaluations with large content space.

3) Parameter Setting: Throughout the experiments, we do not finetune the parameter. On the synthetic setting, we do not intentionally select the synthetic cases where NOC performs better. We mainly use the following configuration: The Zipf parameter is 0.6 among all synthetic traces. The Poisson parameter of the Zipf-Poisson traces is randomly selected from $[10^3, 10^5]$, and the shuffle times of Zipf-Shuffle is uniformly selected from $[0, 2^6]$. We sample 50000 requests of 240 contents in synthetic traces and take 100 requests as an episode. We set the cache capacity $C = 10$. We use hit rate as the performance metric. For fairness, we do not tune the parameter of NOC, and fix $\delta = 0.5$ as the default.

B. Verification of Dynamic Adaptation

1) Static Environment V.s. Dynamic Environment: We first compare the average cache hit rate of NOC and the baseline algorithms with Zipf and Poisson Zipf traces, which indicate the static and continuously dynamic environment respectively. As illustrated in the Fig. 5(a), we observe that in Zipf traces, LFU achieves the best performance and quickly converges to the best, which supports its tight static regret as previously analyzed. NOC, OGA and DRL gets similar final performance with LFU, with around 1% worse. This is because NOC and OGA policies also have sublinear (not tight) static regret, and DRL is designed to maximize the long-term cache hit rate. Since OGA has a fixed stepsize designed for a fixed length T of requests, it shows weaker adaptation during $1 \sim T - 1$. LRU gets the worst hit rate since it has linear static regret [40].

Compared to the good result in static traces, for the dynamic environment with Poisson Zipf traces, NOC achieves the best average hit rate (Fig. 4), which supports its sublinear dynamic regret. Recalling in the previous analysis, LFU, LRU and OGA have linear dynamic regret, and therefore perform $1.5\% \sim 14\%$ worse than NOC. Without a strict theoretical guarantee, sophisticated designed DRL works 2.5% worse than NOC. Notice in Fig. 5(b), due to the relatively stable environment in a short time period, LFU could quickly achieve good results. However, its performance degrades rapidly over time since the popularity distribution changes with time. As NOC can approach the optimal dynamic policy, it thus achieves good results at the beginning and maintains the best performance for a long time. This is consistent with our previous analysis.

2) Impact of Dynamicity: To further evaluate the dynamic adaptation of NOC, we then investigate how the dynamicity affects the performance of caching algorithms by running experiments on the Zipf-Shuffle traces with different shuffle times. According to Fig. 6(a), when the dynamicity increases (the number of shuffles increases), the performance of LFU and OGA decreases significantly, while the NOC policy keeps the best performance (up to 7% hit rate improvement than OGA). The reason is that the static regret could not model dynamic environments, and therefore static regret minimization methods would lose their performance when dynamicity increases, while NOC has sublinear dynamic regret and thus can adapt to environmental changes. The synthesized traces prove that the validation of dynamic adaptation. DRL has almost the same result as NOC, showing that DRL also works well with sudden changes as in previous paper [20].

To see how different algorithms dealing with environmental changes are, we conduct on one Zipf-Shuffle trace with 16 times of shuffles and plot Fig. 6(b). LFU achieves the highest performance before the distribution shuffle, but falls and struggles to recover after shuffling. OGA has better adaptation but is still difficult to adapt to the new environment. The result aligns

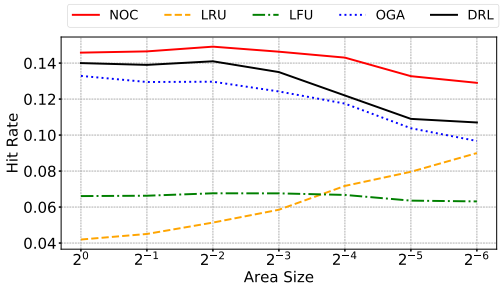


Fig. 7. Average cache hit rate versus the service range of an edge device. The x axis is the ratio of area size with the largest area.

with our theory that LFU and OGA have sublinear static regret and perform well in the static environment, but have linear dynamic regret and work badly with environmental changes. In contrast, NOC with sublinear dynamic regret is close to LFU before shuffling, soon adapts to the environment change, and becomes the best after shuffling, which verifies the dynamic adaptation. We highlight that NOC does not make any prior assumptions, and can fit into any dynamic practical circumstance based on this point. DRL reacts to environmental changes very similar to NOC, but it maintains larger memory and computational cost because of feature memory and the deep network. LRU performs not well all the time, which aligns with its linear dynamic regret analysis.

C. Real-World Traces Results

1) Impact of Edgeness: To evaluate the impact of the "edgeness" on our proposal's performance, we vary the service range of an edge cache device from $15.5 \times 17.4\,\text{km}^2$ to $2.0 \times 2.2\,\text{km}^2$ in the iQIYI dataset. As illustrated in Fig. 7, NOC maintains 1% to 7% outperformance of hit rate compared to baselines from big area to small area size. As the "edgeness" increases, NOC shows more superiority than OGA and DRL, which verifies that our method is suitable on the edge. Recall in our trace measurement, the real-world requested pattern has demonstrated high dynamics and diversities. OGA and LFU are analyzed to have linear dynamic regret, thus are not adaptive to dynamicity on the edge. DRL depends on manual features that cannot model such diverse settings, from which the bias makes the performance worse than NOC. Note that

It should be noted that the curve of LRU (increasing) is different from others (decreasing). This is because the users tend to repeatedly click the same content. For example, it may take several clicks for a user to finish one video, especially for the long-video platform as iQIYI, where the user may interrupt the video for shifting to other apps due to the long watching time. And when the user returns to iQIYI, it makes a new request. Then, the repeatability (that the contents are more likely to be recently requested) is more significant than regularity (that the contents are requested by some popularity pattern) in the edge, leading to the increasing pattern of LRU and decreasing pattern of other learning-based methods. However, since NOC can capture both the repeatability and regularity, it still maintains a more remarkable outperformance than LRU.

TABLE II
AVERAGE HIT RATE (FIRST ROW) AND AVERAGE PERFORMANCE RANK (SECOND ROW) OF RANDOM AREAS IN REAL-WORLD TRACES

Method	**NOC**	LRU	LFU	OGA	DRL
Hit rate	**12.8±0.7%**	11.2±1.2%	6.5±0.5%	9.6±1.0%	**11.7±0.8%**
Rank	**1.15± 0.47**	2.95± 0.86	5.0± 0.0	3.7± 0.45	**2.2± 0.50**

2) Robustness of NOC: We also extract the user traces from the iQIYI dataset in 20 random areas with $2 \times 2\,\text{km}^2$ area size. The request pattern of the traces in each area is considered to vary as the users may have different preferences. We applied NOC on these traces to verify its robustness with different patterns. The results are illustrated in Table II. NOC keeps the best performance in almost all (as shown by the average performance rank in the second row of the table) of these areas, and up to 6.3% (97% improvement) better than LFU, 1.6% (14% improvement) better than LRU, 3.2% (33% improvement) better than OGA, 1.1% (9.4% improvement) better than DRL. Therefore, we claim that NOC achieves at least 9.4% improvement than the baselines. The reason is that the sublinear dynamic regret, which has no prior assumption, guarantees NOC to work well in different areas. This demonstrates NOC's superiority in real-world applications and further supports the claims of robustness and effectiveness. DRL policy shows good results in the setting of [20], and performs better than OGA. However, it is worse than NOC averagely, revealing that this model is not robust and may only work well in specific settings.

V. RELATED WORK

A. Edge Content Caching

Several studies explored building caches at the "edge" of networks, such as cellular base stations [33], [41]. Meanwhile, edge content delivery also has the potential to reduce the end-to-end delay [42]. Based on large-scale real-world traces, Ma *et al.* [8] verified the performance of edge content delivery, even under diverse user mobility patterns. To improve the scalability of edge content delivery, distributed caching has been studied [34]; and due to the limited resources at each edge device, there are also studies focusing on collaborative caching and coded caching strategies for edge content delivery [35]–[38]. Another line of the literature studies the device-to-device content delivery network [45]–[47], where the mobile devices can also act as cache servers. At an edge cache, traditional replacement strategies are usually straightforward, e.g. based on request volume of each content item [15].

B. Popularity Dynamics in Edge Caching

However, the request patterns change significantly at an edge cache device if it only serves a small group of users nearby. Understanding the popularity patterns thus becomes the key factor to improve edge content delivery performance. Traverso *et al.* [14] introduced a Poisson shot noise model to model dynamic popularities, and Qi *et al.* [16] proposed to combine the independent reference model with shot noise model. However,

such popularity predictions are based on the stochastic assumption, which is too strong in real-world systems, e.g. content request patterns can be affected by many other confounding factors like mobility and social influence.

C. Non-Stationary Replacement

1) Reinforcement Learning Based Replacement: Other studies aim to improve the robustness by *model-free* learning algorithms by using RL to capture the changing content popularity. Sadeghi *et al.* [21] proposed to let a cache node to track the space-time popularity dynamics in an online manner. Zhong *et al.* [20] proposed to use a deep RL framework that utilizes Wolpertinger architecture to generate content replacement decisions. Zhang *et al.* [26] proposed to model the caching problem as a *group linear* model. Somuyiwa *et al.* [24], [25] proposed to optimize a parametric policy that determines cache replacement. Zhong *et al.* [4] proposes deep actor-critic reinforcement learning based policies for both centralized and decentralized content caching. Wang *et al.* [37] integrated long short term memory network (LSTM) with the actor-critic model to better handle time series dynamics from the historical requests. However, previous reinforcement learning-based solutions have the following drawbacks: most of them are empirical without a strong theoretical guarantee on performance, and the excellent performance may only stand on specific settings. Thus, they lose the generality of their claims.

2) Online Learning Based Replacement: Online learning provides a design perspective with strong theoretical guarantees and no prior assumption. To address the environmental changes, Muller *et al.* [29]–[31] have proposed a contextual bandit method, in which the agent learned from the time-dependent features and predicted dynamically. However, storing and computing these features would take large memory and computational costs, since the content space is often huge in practice. Tan *et al.* [32] and Garg *et al.* [39] proposed to use online learning to predict content popularity only from the request history, and then made cache complication based on the prediction. However, they only give the bound for accuracy of popularity prediction, and the caching performance can not be guaranteed.

As a study that is the most related to our proposal, [40] is the first to connect the caching problem with online convex optimization to minimize the *static regret*. They provided a sublinear regret, which guaranteed that their online caching strategy performed approximately to the best static policy. An optimal static regret bound was achieved based on a Follow the Perturbed Leader based policy by Bhattacharjee *et al.* [9], which made the convergence faster. However, their solution suffers under dynamic scenarios, since the best static policy becomes worse as the environment changes.

VI. Conclusion

In this paper, we solve the problem of optimal caching in dynamic environments, motivated by our trace-driven study of iQIYI traces, via dynamic online learning. First, we connect the above problem with the dynamic regret minimization problem. Second, we propose a practical cache policy called Nearly Optimal Cache (NOC), and prove it to have a sublinear dynamic regret, which implies a consistent theoretical performance with the optimal cache policy in hindsight. Also, we give linear dynamic regret lower bound for previous methods; as a result, previous methods would be degraded when the environment changes. Finally, we show that NOC works well for cache problems with static pattern and different degrees of dynamicity, which validates the dynamic adaptation as in theory. We also conduct experiments on real-world traces with various area sizes and locations, where NOC keeps its outperformance, supporting the effectiveness and robustness on edge caching tasks.

Appendix
Proof

A. Proof of Theorem 3

Proof: Recall the definition of the dynamic regret

$$\begin{aligned}\text{DA-Regret}_T &= \sum_{t=1}^{T} f_t(\boldsymbol{y}_t(\sigma^*)) - \sum_{t=1}^{T} f_t(\boldsymbol{y}_t)\\ &= \sum_{t=1}^{T}(-f_t(\boldsymbol{y}_t)) - \sum_{t=1}^{T}(-f_t(\boldsymbol{y}_t(\sigma^*))),\end{aligned}$$

where we rearrange it to compare the convex function $-f_t$ (equivalent transformation from maximization to minimization). For simply representation, we denote $g_t = -f_t$ and $\nabla g_t(\boldsymbol{y}_t)$ by ∇_t. By the convexity of g_t (concavity of f_t), we have for $t = 1, \ldots, T$

$$\begin{aligned}&g_t(\boldsymbol{y}_t) - g_t(\boldsymbol{y}_t(\sigma^*))\\ &\le \nabla_t^{\intercal}(\boldsymbol{y}_t - \boldsymbol{y}_t(\sigma^*))\\ &= \nabla_t^{\intercal}(\boldsymbol{y}_{t+1} - \boldsymbol{y}_t(\sigma^*)) + \nabla_t^{\intercal}(\boldsymbol{y}_t - \boldsymbol{y}_{t+1})\\ &= \frac{1}{\eta_t}\eta_t\nabla_t^{\intercal}(\boldsymbol{y}_{t+1} - \boldsymbol{y}_t(\sigma^*)) + \nabla_t^{\intercal}(\boldsymbol{y}_t - \boldsymbol{y}_{t+1})\\ &\le \frac{1}{\eta_t}(\boldsymbol{y}_t - \boldsymbol{y}_{t+1})^{\intercal}(\boldsymbol{y}_{t+1} - \boldsymbol{y}_t(\sigma^*)) + \nabla_t^{\intercal}(\boldsymbol{y}_t - \boldsymbol{y}_{t+1}).\end{aligned}$$

The last inequality comes from the OGD iteration, the range of stepsize $\eta_t > 0$, and projection property on convex set (i.e. $\mathcal{K}_C$). The reason is that

$$\begin{aligned}&(\boldsymbol{y}_t - \boldsymbol{y}_{t+1})^{\intercal}(\boldsymbol{y}_{t+1} - \boldsymbol{y}_t(\sigma^*))\\ &= (\boldsymbol{y}_t - \Pi_{\mathcal{K}_C}(\boldsymbol{y}_t - \eta_t\nabla_t))^{\intercal}(\boldsymbol{y}_{t+1} - \boldsymbol{y}_t(\sigma^*))\\ &= (\boldsymbol{y}_t - \boldsymbol{y}_t + \eta_t\nabla_t)^{\intercal}(\boldsymbol{y}_{t+1} - \boldsymbol{y}_t(\sigma^*))\\ &\quad + (\boldsymbol{y}_t - \eta_t\nabla_t - \Pi_{\mathcal{K}_C}(\boldsymbol{y}_t - \eta_t\nabla_t))^{\intercal}(\boldsymbol{y}_{t+1} - \boldsymbol{y}_t(\sigma^*))\\ &\ge \eta_t\nabla_t^{\intercal}(\boldsymbol{y}_{t+1} - \boldsymbol{y}_t(\sigma^*)).\end{aligned}$$

We then decompose $(\boldsymbol{y}_t - \boldsymbol{y}_{t+1})^{\intercal}(\boldsymbol{y}_{t+1} - \boldsymbol{y}_t(\sigma^*))$ into

$$\begin{aligned}&(\boldsymbol{y}_t - \boldsymbol{y}_{t+1})^{\intercal}(\boldsymbol{y}_{t+1} - \boldsymbol{y}_t(\sigma^*))\\ &= \boldsymbol{y}_t^{\intercal}\boldsymbol{y}_{t+1} - \boldsymbol{y}_t^{\intercal}\boldsymbol{y}_t(\sigma^*) - \boldsymbol{y}_{t+1}^{\intercal}\boldsymbol{y}_{t+1} + \boldsymbol{y}_{t+1}^{\intercal}\boldsymbol{y}_t(\sigma^*)\\ &= \frac{1}{2}(\|\boldsymbol{y}_t(\sigma^*) - \boldsymbol{y}_t\|^2 - \|\boldsymbol{y}_{t+1} - \boldsymbol{y}_t\|^2 - \|\boldsymbol{y}_t(\sigma^*) - \boldsymbol{y}_{t+1}\|^2).\end{aligned}$$

Now, we have

$$\begin{aligned}
&g_t(\boldsymbol{y}_t(\sigma^*)) - g_t(\boldsymbol{y}_t)\\
&\le \frac{1}{2\eta_t}(\|\boldsymbol{y}_t(\sigma^*) - \boldsymbol{y}_t\|^2 - \|\boldsymbol{y}_{t+1} - \boldsymbol{y}_t\|^2 - \|\boldsymbol{y}_t(\sigma^*) - \boldsymbol{y}_{t+1}\|^2)\\
&\quad + \nabla_t^\intercal(\boldsymbol{y}_t - \boldsymbol{y}_{t+1})\\
&= \frac{1}{2\eta_t}\|\boldsymbol{y}_t(\sigma^*) - \boldsymbol{y}_t\|^2 - \frac{1}{2\eta_t}\|\boldsymbol{y}_t(\sigma^*) - \boldsymbol{y}_{t+1}\|^2\\
&\quad - \frac{1}{2\eta_t}\|\boldsymbol{y}_{t+1} - \boldsymbol{y}_t\|^2 + \nabla_t^\intercal(\boldsymbol{y}_t - \boldsymbol{y}_{t+1}).
\end{aligned}$$

Since, we know that

$$\begin{aligned}
&\frac{1}{2\eta_t}\|\boldsymbol{y}_t(\sigma^*) - \boldsymbol{y}_t\|^2 - \frac{1}{2\eta_t}\|\boldsymbol{y}_t(\sigma^*) - \boldsymbol{y}_{t+1}\|^2\\
&= \frac{1}{2\eta_t}(\|\boldsymbol{y}_t(\sigma^*) - \boldsymbol{y}_t\|^2 - \|\boldsymbol{y}_{t+1}(\sigma^*) - \boldsymbol{y}_{t+1}\|^2)\\
&\quad + \frac{1}{2\eta_t}(\|\boldsymbol{y}_{t+1}(\sigma^*) - \boldsymbol{y}_{t+1}\|^2 - \|\boldsymbol{y}_t(\sigma^*) - \boldsymbol{y}_{t+1}\|^2).
\end{aligned}$$

Then, we have

$$\begin{aligned}
&g_t(\boldsymbol{y}_t(\sigma^*)) - g_t(\boldsymbol{y}_t)\\
&\le \frac{1}{2\eta_t}(\|\boldsymbol{y}_t(\sigma^*) - \boldsymbol{y}_t\|^2 - \|\boldsymbol{y}_{t+1}(\sigma^*) - \boldsymbol{y}_{t+1}\|^2) \qquad (2)\\
&\quad - \frac{1}{2\eta_t}\|\boldsymbol{y}_{t+1} - \boldsymbol{y}_t\|^2 + \nabla_t^\intercal(\boldsymbol{y}_t - \boldsymbol{y}_{t+1}) \qquad (3)\\
&\quad + \frac{1}{2\eta_t}(\|\boldsymbol{y}_{t+1}(\sigma^*) - \boldsymbol{y}_{t+1}\|^2 - \|\boldsymbol{y}_t(\sigma^*) - \boldsymbol{y}_{t+1}\|^2). \qquad (4)
\end{aligned}$$

Now, we consider to bound the above three terms step by step. First, for equation (2)

$$\begin{aligned}
&\sum_{t=1}^{T}\frac{1}{2\eta_t}(\|\boldsymbol{y}_t(\sigma^*) - \boldsymbol{y}_t\|^2 - \|\boldsymbol{y}_{t+1}(\sigma^*) - \boldsymbol{y}_{t+1}\|^2)\\
&\le \frac{1}{2}\sum_{t=1}^{T}\left(\frac{1}{\eta_t} - \frac{1}{\eta_{t-1}}\right)\|\boldsymbol{y}_t(\sigma^*) - \boldsymbol{y}_t\|^2\\
&\le \frac{1}{2\eta_T}\max(\|\boldsymbol{y}_t(\sigma^*) - \boldsymbol{y}_t\|^2)\\
&\le \frac{C}{\eta_T}.
\end{aligned}$$

The last equation is from our definition of $\mathcal{K}_C$. Second, for equation (3), we have

$$\begin{aligned}
&\nabla_t^\intercal(\boldsymbol{y}_t - \boldsymbol{y}_{t+1}) - \frac{1}{2\eta_t}\|\boldsymbol{y}_{t+1} - \boldsymbol{y}_t\|^2\\
&= \frac{1}{2\eta_t}(2\eta_t\nabla_t^\intercal(\boldsymbol{y}_t - \boldsymbol{y}_{t+1}) - \|\boldsymbol{y}_{t+1} - \boldsymbol{y}_t\|^2)\\
&= \frac{1}{2\eta_t}(-\|\boldsymbol{y}_t - \eta_t\nabla_t - \boldsymbol{y}_{t+1}\|^2 + \eta_t^2\|\nabla_t\|^2)\\
&= \frac{1}{2\eta_t}(-\|\boldsymbol{y}_{t+1} - \boldsymbol{y}_{t+1}\|^2 + \eta_t^2\|\nabla_t\|^2)\\
&= \frac{\eta_t}{2}\|\nabla_t\|^2.
\end{aligned}$$

The last two inequality comes from the Pythagorean Theorem [43]. Therefore,

$$\begin{aligned}
&\sum_{t=1}^{T}(\nabla_t^\intercal(\boldsymbol{y}_t - \boldsymbol{y}_{t+1}) - \frac{1}{2\eta_t}\|\boldsymbol{y}_{t+1} - \boldsymbol{y}_t\|^2)\\
&\le \sum_{t=1}^{T}\frac{\eta_t}{2}\|\nabla_t\|^2\\
&\le \sum_{t=1}^{T}\frac{\eta_t}{2}G^2.
\end{aligned}$$

Third, for equation (4), we have

$$\begin{aligned}
&\sum_{t=1}^{T}\frac{1}{2\eta_t}(\|\boldsymbol{y}_{t+1}(\sigma^*) - \boldsymbol{y}_{t+1}\|^2 - \|\boldsymbol{y}_t(\sigma^*) - \boldsymbol{y}_{t+1}\|^2)\\
&\le \sum_{t=1}^{T}\frac{1}{2\eta_t}\|\boldsymbol{y}_{t+1}(\sigma^*) - \boldsymbol{y}_t(\sigma^*)\|^2\\
&\le \frac{1}{2\eta_T}\sum_{t=1}^{T}\|\boldsymbol{y}_{t+1}(\sigma^*) - \boldsymbol{y}_t(\sigma^*)\|^2\\
&= \frac{V(T)}{2\eta_T}.
\end{aligned}$$

Conbiming the above bounds for the three terms, we can finally obtain

$$\text{DA-Regret}_T \le \frac{C}{\eta_T} + \frac{V(T)}{2\eta_T} + \sum_{t=1}^{T}\frac{\eta_t}{2}G^2.$$

■

B. Proof of Theorem 4

Proof: According to Theorem 3, we have

$$\begin{aligned}
&\text{DA-Regret}_T(\text{NOC}) \le \frac{C}{\eta_T} + \frac{V(T)}{2\eta_T} + \frac{G^2}{2}\sum_{t=1}^{T}\eta_t\\
&= CT^{(1-\delta)/2} + \frac{1}{2}V(T)T^{(1-\delta)/2} + \frac{G^2}{2}\sum_{t=1}^{T}\frac{1}{t^{(1-\delta)/2}}\\
&\le CT^{(1-\delta)/2} + V(T)T^{(1-\delta)/2} + \frac{G^2}{2}\int_1^{T+1}\frac{1}{t^{(1-\delta)/2}}\\
&= CT^{(1-\delta)/2} + V(T)T^{(1-\delta)/2} + \frac{G^2}{1+\delta}t^{(1+\delta)/2}\Big|_1^{T+1}\\
&\le CT^{(1-\delta)/2} + V(T)T^{(1-\delta)/2} + \frac{G^2}{1+\delta}T^{(1+\delta)/2}\\
&\le O(T^{(1+\delta)/2}).
\end{aligned}$$

■

C. Proof of Theorem 5

Proof: Creating an example to derive the worst case regret is typical lower bound the online algorithm. Here, we aim to create such an example where the content with the highest frequency (calculated from the beginning) is not popular in the latter part of the time. Thus, the LFU algorithm that depends on the frequency signal will fail in that case. Assume that we have a request sequence as $\{1, 1, \ldots, 1, 2, 2, \ldots, 2\}$, which includes $\lfloor \frac{T}{2} \rfloor$ requests to content item 1 and $\lceil \frac{T}{2} \rceil$ requests to content item 2. Suppose the cache size is only $C = 1$, and the optimal cache is allowed to make $V(T) = 1$ time of content replacement, and the performance function is still the hit rate $f_t(\boldsymbol{y}_t) = \boldsymbol{x}_t^\intercal \boldsymbol{y}_t$. Hence, the optimal caching policy should cache content 1 for the first $\lfloor \frac{T}{2} \rfloor$ requests, and replaces it with content 2 for the next $\lceil \frac{T}{2} \rceil$ requests; and it thus achieves 100% hit rate, while LFU can only hit the first $\lfloor \frac{T}{2} \rfloor$ requests, as the frequence of content 2 will never be larger than the frequence of content 1 during the last $\lceil \frac{T}{2} \rceil$ requests. Thus, we conclude that under the worst case, the dynamic regret for LFU will be

$$\text{DA-Regret}_T(\text{LFU}) \geq T - \left\lfloor \frac{T}{2} \right\rfloor = \left\lceil \frac{T}{2} \right\rceil = O(T).$$

■

D. Proof of Theorem 6

Proof: Here, we aim to create such an example where the recently requested content will not be requested in the next. Thus, the LRU algorithm that depends on the recency signal will fail in that case. Assume that we have a request sequence $\{1, 2, 1, 2, \ldots, 1, 2\}$, which includes $\lceil \frac{T}{2} \rceil$ requests to content item 1 and $\lfloor \frac{T}{2} \rfloor$ requests to content item 2. Suppose the cache size is $C = 1$ and the optimal cache is allowed to make content set change by $V(T) = 1$ time, and the performance function is also the hit rate $f_t(\boldsymbol{y}_t) = \boldsymbol{x}_t^\intercal \boldsymbol{y}_t$. Hence, the optimal cache achieves 50% hit rate (hit $\lceil \frac{T}{2} \rceil$ times after the first content change to either content 1 or 2), and LRU misses all the requests after the first request (with no content change limitation). Therefore the dynamic regret has the below lower bound.

$$\text{DA-Regret}_T(\text{LRU}) \geq O(T) = \left\lceil \frac{T}{2} \right\rceil - 0 = \left\lceil \frac{T}{2} \right\rceil = O(T).$$

■

E. Proof of Theorem 7

Proof: Here, we aim to create such an example where learning step size of OGA [40] is too small to adapt to the shifting request pattern. Thus, the OGA algorithm will fail in that case. Assume that we have a request sequence $\{1, \ldots, 1, 2, \ldots, 2, 3, \ldots, 3, 4, \ldots, \lfloor \sqrt{T} \rfloor\}$, which includes $\lfloor \sqrt{T} \rfloor$ requests for each content $n = 1, 2, \ldots, \lfloor \sqrt{T} \rfloor$ (T requests in total). Suppose the cache size is $C = 1$ and the optimal cache is allowed to make $V(T) = \sqrt{T}$ times of replacement, and the performance function is hit rate $f_t(\boldsymbol{y}_t) = \boldsymbol{x}_t^\intercal \boldsymbol{y}_t$. Hence, the dynamic optimal cache achieves 100% hit rate (as it caches the $\lfloor \sqrt{T} \rfloor$ new requested contents in advance). On the other hand, the OGA policy can hit no more than $\lfloor (1 - \frac{1}{2\sqrt{2}})T \rfloor$ requests. By the definition of [40], when receiving a new requested content i ($i = 1, \ldots, \lfloor \sqrt{T} \rfloor$), it then takes more than $\lceil \frac{\sqrt{T}}{2\sqrt{2}} \rceil$ iterations to replace the previous cached content by the new one. Therefore the OGA policy will miss more than $\lfloor \frac{\sqrt{T}}{2\sqrt{2}} \rfloor \lfloor \sqrt{T} \rfloor = O(T)$ requests. Thus, we have the dynamic regret low bound for OGA:

$$\text{DA-Regret}_T(\text{OGA}) \geq O(T).$$

■

F. Proof of Theorem 8

Proof: We create the same situation as in the proof of LFU. Hence, the optimal caching policy achieves 100% hit rate, while FTPL has no more than $1/2$ popularity to hit the last $\frac{T}{2}$ requests, as the frequence of content 2 will never be larger than the frequence of content 1 during the last $\frac{T}{2}$ requests. Thus, we conclude that under the worst case, the expected dynamic regret for FTPL will be

$$E[\text{DA-Regret}_T(\text{FTPL})] \geq T - \frac{T}{2} - \frac{T}{2} \cdot \frac{1}{2} = \frac{T}{4} = O(T).$$

■

REFERENCES

[1] E. J. O'neil, P. E. O'neil, and G. Weikum, "The LRU-K page replacement algorithm for database disk buffering," *Acm Sigmod Rec.*, vol. 22, no. 2, pp. 297–306, 1993.

[2] L. Maggi, L. Gkatzikis, G. Paschos, and J. Leguay, "Adapting caching to audience retention rate," *Comput. Commun.*, vol. 116, pp. 159–171, 2018.

[3] K. Shanmugam, N. Golrezaei, A. G. Dimakis, A. F. Molisch, and G. Caire, "FemtoCaching: Wireless video content delivery through distributed caching helpers," in *Proc. INFOCOM*. IEEE, 2012, pp. 1107–1115.

[4] C. Zhong, M. C. Gursoy, and S. Velipasalar, "Deep reinforcement learning-based edge caching in wireless networks," *IEEE Trans. Cogn. Commun. Netw.*, vol. 6, no. 1, pp. 48–61, Mar. 2020.

[5] A. V. Aho, P. J. Denning, and J. D. Ullman, "Principles of optimal page replacement," *J. ACM (JACM)*, vol. 18, no. 1, pp. 80–93, 1971.

[6] G. Casale, "Analyzing replacement policies in list-based caches with non-uniform access costs," in *Proc. IEEE Conf. Comput. Commun.*, 2018, pp. 432–440.

[7] D. Lee *et al.*, "LRFU: A spectrum of policies that subsumes the least recently used and least frequently used policies," *IEEE Trans. Comput.*, vol. 50, no. 12, pp. 1352–1361, Dec. 2001.

[8] G. Ma, Z. Wang, M. Zhang, J. Ye, M. Chen, and W. Zhu, "Understanding performance of edge content caching for mobile video streaming," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 5, pp. 1076–1089, May 2017.

[9] R. Bhattacharjee, S. Banerjee, and A. Sinha, "Fundamental limits on the regret of online network-caching," in *Proc. Abstr. SIGMETRICS/Perform. Joint Int. Conf. Meas. Model. Comput. Syst.*, 2020, pp. 15–16.

[10] W. Hu, Z. Wang, M. Ma, and L. Sun, "Edge video CDN: A Wi-Fi content hotspot solution," *J. Comput. Sci. Technol.*, vol. 31, no. 6, pp. 1072–1086, 2016.

[11] C. Fricker, P. Robert, and J. Roberts, "A versatile and accurate approximation for LRU cache performance," in *Proc. 24th Int. Teletraffic Congr.*, 2012, pp. 1–8.

[12] D. D. Sleator and R. E. Tarjan, "Amortized efficiency of list update and paging rules," *Commun. ACM*, vol. 28, no. 2, pp. 202–208, 1985.

[13] T. Cisco, "Cisco visual networking index: Global mobile data traffic forecast update, 2012-2017," *Cisco Public Inf.*, vol. 26, p. 27, 2013.

[14] S. Traverso *et al.*, "Temporal locality in today's content caching: Why it matters and how to model it," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 5, pp. 5–12, 2013.

[15] M. Leconte, G. Paschos, L. Gkatzikis, M. Draief, S. Vassilaras, and S. Chouvardas, "Placing dynamic content in caches with small population," in *Proc. IEEE INFOCOM 35th Annu. Int.Conf. Comput. Commun.*, 2016, pp. 1–9.

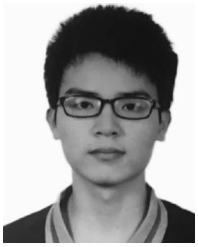
[16] K. Qi, S. Han, and C. Yang, "Learning a hybrid proactive and reactive caching policy in wireless edge under dynamic popularity," *IEEE Access*, vol. 7, pp. 120788–120801, 2019.
[17] P. Cheng *et al.*, "Localized small cell caching: A machine learning approach based on rating data," *IEEE Trans. Commun.*, vol. 67, no. 2, pp. 1663–1676, Feb. 2019.
[18] S. Mehrizi, A. Tsakmalis, S. Chatzinotas, and B. Ottersten, "A Bayesian poisson-Gaussian process model for popularity learning in edge-caching networks," *IEEE Access*, vol. 7, pp. 92341–92354, 2019.
[19] S. Mehrizi, A. Tsakmalis, S. Chatzinotas, and B. Ottersten, "A feature-based Bayesian method for content popularity prediction in edge-caching networks," in *Proc. IEEE Wireless Commun. Netw. Conf.*, 2019, pp. 1–6.
[20] C. Zhong, M. C. Gursoy, and S. Velipasalar, "A deep reinforcement learning-based framework for content caching," in *Proc. 52nd Annu. Conf. Inf. Sci. Syst.*, 2018, pp. 1–6.
[21] A. Sadeghi, F. Sheikholeslami, and G. B. Giannakis, "Optimal and scalable caching for 5G using reinforcement learning of space-time popularities," *IEEE J. Sel. Topics Signal Process.*, vol. 12, no. 1, pp. 180–190, Feb. 2018.
[22] K. Guo, C. Yang, and T. Liu, "Caching in base station with recommendation via Q-learning," in *Proc. IEEE Wireless Commun. Netw. Conf.*, 2017, pp. 1–6.
[23] A. Sadeghi, F. Sheikholeslami, and G. B. Giannakis, "Optimal dynamic proactive caching via reinforcement learning," in *Proc. IEEE 19th Int. Workshop Signal Process. Adv. Wireless Commun.*, 2018, pp. 1–5.
[24] S. O. Somuyiwa, A. György, and D. Gündüz, "A reinforcement-learning approach to proactive caching in wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 6, pp. 1331–1344, Jun. 2018.
[25] S. O. Somuyiwa, D. Gündüz, and A. Gyorgy, "Reinforcement learning for proactive caching of contents with different demand probabilities," in *Proc. 15th Int. Symp. Wireless Commun. Syst.*, 2018, pp. 1–6.
[26] N. Zhang, K. Zheng, and M. Tao, "Using grouped linear prediction and accelerated reinforcement learning for online content caching," in *Proc. IEEE Int. Conf. Commun. Workshops*, 2018, pp. 1–6.
[27] N. Garg, M. Sellathurai, and T. Ratnarajah, "Content placement learning for success probability maximization in wireless edge caching networks," in *Proc. ICASSP IEEE Int. Conf. Acoust., Speech Signal Process.*, 2019, pp. 3092–3096.
[28] P. Blasco and D. Gündüz, "Learning-based optimization of cache content in a small cell base station," in *Proc. IEEE Int. Conf. Commun.*, 2014, pp. 1897–1903.
[29] S. Müller, O. Atan, M. van der Schaar, and A. Klein, "Smart caching in wireless small cell networks via contextual multi-armed bandits," in *Proc. IEEE Int. Conf. Commun.*, 2016, pp. 1–7.
[30] S. Müller, O. Atan, M. van der Schaar, and A. Klein, "Context-aware proactive content caching with service differentiation in wireless networks," *IEEE Trans. Wireless Commun.*, vol. 16, no. 2, pp. 1024–1036, Feb. 2017.
[31] S. Li, J. Xu, M. van der Schaar, and W. Li, "Trend-aware video caching through online learning," *IEEE Trans. Multimedia*, vol. 18, no. 12, pp. 2503–2516, Dec. 2016.
[32] Y. Tan, Y. Yuan, T. Yang, and B. Hu, "Learning-based caching with unknown popularity in wireless video networks," in *Proc. IEEE 85th Veh. Technol. Conf.*, 2017, pp. 1–5.
[33] N. Golrezaei, K. Shanmugam, A. G. Dimakis, and A. F. Molisch, "FemtoCaching: Wireless video content delivery through distributed caching helpers," in *Proc. IEEE Conf. Comput. Commun.*, 2012, pp. 1107–1115.
[34] K. Poularakis, G. Iosifidis, A. Argyriou, I. Koutsopoulos, and L. Tassiulas, "Caching and operator cooperation policies for layered video content delivery," in *Proc. IEEE Int. Conf. Comput. Commun.*, 2016, pp. 1–9.
[35] J. Hachem, N. Karamchandani, and S. Diggavi, "Content caching and delivery over heterogeneous wireless networks," in *Proc. IEEE Int. Conf. Comput. Commun.*, 2015, pp. 756–764.
[36] M. A. Maddah-Ali and U. Niesen, "Fundamental limits of caching," *IEEE Trans. Inf. Theory*, vol. 60, no. 5, pp. 2856–2867, May 2014.
[37] F. Wang, F. Wang, J. Liu, R. Shea, and L. Sun, "Intelligent video caching at network edge: A multi-agent deep reinforcement learning approach," in *Proc. IEEE Conf. Comput. Commun.*, 2020, pp. 2499–2508.
[38] H. Pang, J. Liu, X. Fan, and L. Sun, "Toward smart and cooperative edge caching for 5G networks: A deep learning based approach," in *Proc. IEEE/ACM 26th Int. Symp. Qual. Serv.*, 2018, pp. 1–6.
[39] N. Garg, M. Sellathurai, V. Bhatia, B. Bharath, and T. Ratnarajah, "Online content popularity prediction and learning in wireless edge caching," *IEEE Trans. Commun.*, vol. 68, no. 2, pp. 1087–1100, Feb. 2020.
[40] G. S. Paschos, A. Destounis, L. Vigneri, and G. Iosifidis, "Learning to cache with no regrets," in *Proc. IEEE Conf. Comput. Commun.*, 2019, pp. 235–243.
[41] J. Roberts and N. Sbihi, "Exploring the memory-bandwidth tradeoff in an information-centric network," in *Proc. Int. Teletraffic Congr.*, 2013, pp. 1–9.
[42] H. Ahlehagh and S. Dey, "Video-aware scheduling and caching in the radio access network," *IEEE/ACM Trans. Netw.*, vol. 22, no. 5, pp. 1444–1462, Oct. 2014.
[43] E. Hazan *et al.*, "Introduction to online convex optimization," *Found. Trends Optim.*, vol. 2, no. 3/4, pp. 157–325, 2016.
[44] M. Zinkevich, "Online convex programming and generalized infinitesimal gradient ascent," in *Proc. 20th Int. Conf. Mach. Learn.*, 2003, pp. 928–936.
[45] N. Golrezaei, A. G. Dimakis, and A. F. Molisch, "Wireless device-to-device communications with distributed caching," in *Proc. IEEE Int. Symp. Inf. Theory Proc.*, 2012, pp. 2781–2785.
[46] M. Ji, G. Caire, and A. F. Molisch, "Wireless device-to-device caching networks: Basic principles and system performance," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 1, pp. 176–189, Jan. 2016.
[47] J. Sung and D. Lee, "Efficient design and control for network-assisted device-to-device content delivery network," *IEEE Trans. Multimedia*, vol. 23, pp. 2442–2456, 2020.

Shiji Zhou received the bachelor's degree (Hua Luogeng Mathematics Honors Class) from the Chinese Academy of Science, Beihang University, Beijing, China. He is currently working toward the Ph.D. degree with Tsinghua University, Beijing, China. His research interests include non-stationary online learning, domain adaptation, and multimedia networks.

Zhi Wang (Member, IEEE) is currently an Associate Professor with Tsinghua Shenzhen International Graduate School, Shenzhen, China. His research interests include multimedia networks, mobile cloud computing, and large-scale machine learning systems. His research has been covered by prestigious media, including MIT Technology Review. He was the recipient of the Outstanding Doctoral Dissertation Award from China Computer Federation in 2014, Best Paper Award at ACM Multimedia 2012, and Best Student Paper Award at MMM 2015. He was the recipient of the Second Prize of National Natural Science Award and First Prize of Natural Science Award of Ministry of Education in 2017. He is an Associate Editor for the IEEE TRANSACTIONS ON MULTIMEDIA and the Guest Editor of ACM TIST and JCST.

Chenghao Hu received the bachelor's degree from the South China University of Technology, Guangzhou, China, and the master's degree from Tsinghua University, Beijing, China. He is currently working toward the Ph.D. degree with the University of Toronto, Toronto, ON, Canada. His research focuses on distributed machine learning, including efficient training and deployment.

Yinan Mao received the B.Eng. degree in aircraft control and information engineering from Beihang University, Beijing, China, in 2021. He is currently working toward the M.Eng. degree with Tsinghua Shenzhen International Graduate School, Shenzhen, China. His research interests include non-stationary reinforcement learning and multimedia network.



Haopeng Yan received the B.S. degree in computer science and technology from Zhejiang University, Zhejiang, China, in 2018. He is currently working toward the M.S. degree in computer technology with the Department of Computer Science and Technology, Tsinghua University, Beijing, China. His research interests include multimedia network and deep reinforcement learning.

Shanghang Zhang (Member, IEEE) received the master's from Peking University, Beijing, China, and the Ph.D. degree from Carnegie Mellon University, Pittsburgh, PA, USA. She is currently a Postdoc Research Fellow with the Berkeley AI Research Lab (BAIR), EECS, University of California, Berkeley, Berkeley, CA, USA. She is the chief author and editor of book *Deep Reinforcement Learning: Fundamentals, Research and Applications published by Springer Nature*. Her research interests include multimedia intelligence and machine learning, especially sample efficient learning, as reflected in her publications on top-tier journals and conferences, including TMM, TNNLS, NeurIPS, ICLR, ACM MM, CVPR, ICCV, and AAAI. She was the recipient of the Best Paper Award at AAAI 2021, 2018 Rising Stars in EECS, USA, and Qualcomm Innovation Fellowship (QInF) Finalist Award. Dr. Zhang is the chief organizer of several workshops on ICML/NeurIPS, and the special issue on ICMR.

Chuan Wu (Senior Member, IEEE) received the B.Engr. and M.Engr. degrees from the Department of Computer Science and Technology, Tsinghua University, China, in 2000 and 2002, respectively, and the Ph.D. degree from the Department of Electrical and Computer Engineering, University of Toronto, Toronto, ON, Canada, in 2008. Between 2002 and 2004, She was with the Information Technology Industry in Singapore. Since September 2008, she has been with the Department of Computer Science, University of Hong Kong, Hong Kong, where she is currently a Professor. Her current research interests include cloud computing, distributed machine learning systems and algorithms, and intelligent elderly care technologies. She is a Member of ACM, and was the Chair of the Interest Group on Multimedia services and applications over Emerging Networks (MEN) of the IEEE Multimedia Communication Technical Committee (MMTC) from 2012 to 2014. She is an Associate Editor for the IEEE TRANSACTIONS ON CLOUD COMPUTING, IEEE TRANSACTIONS ON MULTIMEDIA, *ACM Transactions on Modeling and Performance Evaluation of Computing Systems,* and IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY. She was also a TPC members and reviewers for various international conferences and journals. She was the co-recipient of the best paper awards of HotPOST 2012 and ACM e-Energy 2016.

Wenwu Zhu (Fellow, IEEE) is currently a Professor with the Computer Science Department, Tsinghua University, Beijing, China, and the Vice Dean of National Research Center on Information Science and Technology. Prior to his current post, he was a Senior Researcher and Research Manager with Microsoft Research Asia. From 2004 to 2008, he was a Chief Scientist and the Director with Intel Research China. During 1996–1999, he was with Bell Labs New Jersey, as a Member of Technical Staff.His current research interests include cross-media Big Data and intelligence, and multimedia edge computing. From January 1, 2017 to December 31, 2019, he was the Editor-in-Chief of the IEEE TRANSACTIONS ON MULTIMEDIA. Since January 1, 2020, he has been the Chair of the steering committee for IEEE TMM and Vice EiC of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY. He was the recipient of the nine best paper awards. He is an AAAS Fellow, SPIE Fellow and a Member of the European Academy of Sciences (Academia Europaea).