Regret-Optimal Learning for Minimizing Edge Caching Service Costs

Guocong Quan
The Ohio State University
Columbus, USA
quan.72@osu.edu

Atilla Eryilmaz
The Ohio State University
Columbus, USA
eryilmaz.2@osu.edu

Ness Shroff
The Ohio State University
Columbus, USA
shroff.11@osu.edu

***Abstract*—Edge caching has been widely implemented to efficiently serve data requests from end users. Numerous edge caching policies have been proposed to adaptively update cache content based on various statistics including data popularities and miss costs. Nevertheless, these policies typically assume that the miss cost for each data item is known, which is not true in real systems. A promising approach would be to use online learning to estimate these unknown miss costs. However, existing techniques cannot be directly applied, because the caching problem has additional cache capacity and cache update constraints that are not covered in traditional learning settings. In this work, we resolve these issues by developing a novel edge caching policy that learns uncertainty miss costs efficiently, and is shown to be asymptotically optimal. We first derive an asymptotic lower bound on the achievable regret. We then design a Kullback-Leibler lower confidence bound (KL-LCB) based edge caching policy, which adaptively learns the random miss costs by following the "optimism in the face of uncertainty" principle. By employing a novel analysis that accounts for the new constraints and the dynamics of the setting, we prove that the regret of the proposed policy matches the regret lower bound, thus showing asymptotic optimality. Further, via numerical experiments we demonstrate the performance improvements of our policy over natural benchmarks.**

***Index Terms*—edge caching, online learning**

I. Introduction

Edge caching has been widely implemented to store data items closer to end users and accelerate data access. It is reported that about 50% photo traffic in Facebook are served by geographically distributed edge caches [1]. Edge caches typically have limited capacity and can only accommodate a small fraction of the entire dataset. When the requested data item is not stored in the edge cache, we call it a cache miss and the data item has to be fetched from the backend data storage to serve the request, which will incur a large delay or cost. A critical question for caching design is *which data items should be stored in the edge cache?*

Numerous caching policies have been designed to update the cache content based on different data statistics. One critical statistic is the miss cost, i.e., the cost to fetch a data item from the backend when it is missed by the edge cache. The miss cost is a general concept depending on the specific application (e.g., the miss cost could represent the latency or bandwidth consumed for fetching the missed data). Intuitively, we can cache data items that may potentially incur larger miss costs, so that the expected cost to serve the request is minimized. By following this principle, various caching policies have been proposed [2], [3], [4], [5], [6]. However, almost all of them assume that the miss costs are known, which is not the case in real systems. The miss cost of a data item could be *random and unknown* in real systems. For example, the miss cost may depend on the geographic locations of the backend storage that sends the missed data back, communication environments, network traffic flows, etc. Existing caching policies cannot satisfactorily handle such uncertainty. To fill this gap, we develop new edge caching policies that learn the uncertain miss costs adaptively and efficiently.

A promising approach is to use online learning to estimate these unknown miss costs. However, existing online learning approaches cannot be directly applied due to the following reasons. 1) The learning actions and the caching decisions are correlated and should be jointly optimized. Specifically, we can observe samples for the uncertain miss costs, only when the corresponding data item is not stored in the edge cache. 2) Caching problems have additional cache capacity and content update constraints that are not covered in the traditional online learning settings. For example, the cache content will remain the same, if there are no cache updates, which naturally introduces time correlations. Due to the dependency between learning and caching decisions, such time correlations will exist in the action sets of the learning process. These constraints make the problem highly non-trivial. We show in Section III that a heuristic design could almost always make wrong caching decisions and achieves poor performance.

We address these challenges by designing a novel edge caching policy that learns the uncertain miss costs efficiently. In particular, we first characterize the best achievable caching performance by establishing a regret lower bound. Inspired by the "optimism in the face of uncertainty" principle in learning literature [7], [8], we then propose a novel KL-LCB based edge caching policy that adaptively learns the unknown miss costs. In order to analyze the theoretical performance of the proposed policy, we are required to prove almost-sure convergence results for critical caching statistics, which are not covered by traditional online learning analysis. Based on these

This work was supported by the NSF grants 2112471, CNS-1717060, CNS-1955535, CNS-2106933, CNS-NeTS-2007231, CNS-NeTS-2106679, CNS-SpecEES-1824337, the ARO grant W911NF2110244 and the ONR grant N00014-19-1-2621.

new results, we prove that the proposed policy achieves the regret lower bound, and is therefore asymptotically optimal. Our key contributions are summarized as follows.

- We reveal the non-triviality of learning miss costs in caching systems. Carefully-designed examples are provided to show that a heuristic learning design could achieve significant inefficiency (see Section III).
- We derive a regret lower bound for any "good" polices (see Section IV), and develop an asymptotically optimal KL-LCB based edge caching policy that achieves this regret lower bound (see Section V). The analysis for the proposed policy employs novel ideas to deal with the new constraints and dynamics in caching systems, and could be potentially leveraged to analyze learning mechanisms for other systems.
- We conduct extensive numerical experiments to evaluate the proposed KL-LCB based edge caching policy, and compare it with a few benchmarks. It is shown that the proposed policy achieves significantly better performance than the other benchmarks (see Section VI).

Related Works: Cost-based caching policies have been extensively studied. The GreedyDual policy evicts the data item with the smallest miss cost when the cache is full [5]. Different designs have been proposed to implement the Greedy-Dual policy [4], [3]. The GreedyDual-Size policy considers data items with different sizes and uses costs per unit data size as a critical factor for caching update [9]. It is further generalized as the Greedy-Dual-Size-Frequency (GDSF) policy to make caching decisions based on the joint effect of data frequency, sizes and costs [2]. Specifically, the GDSF policy attempts to cache the data items with large *frequency* $\times$ *cost / size* values. In [6], Hyperbolic caching is proposed to provide flexible caching service for web applications, and is implemented in real systems such as Redis and Django. It prioritizes data items based on a general function that could depend on miss costs, expiration times, windowing, etc. Other factors (e.g., freshness, latency) are also considered in cost-based policies designed for a variety of applications [10], [11], [12]. Notably, these works assume that the miss costs are known. For unknown miss costs, efficient cost-learning mechanisms jointly optimized with caching decisions are needed.

Leveraging online learning techniques to improve caching performance has received more and more attention. However, most of the existing works in this area focus on learning unknown data popularities or user preference [13], [14], [15], [16], [17], [18]. Learning data popularities has different constraints and dynamics from learning miss costs. And therefore, these approaches cannot be extended to directly solve the cost-learning problem. In [19], fetching costs are considered for caching at small base stations. The paper first assumes known cost distributions and develop efficient algorithms to solve the cost minimization problem. Then, unknown cost distributions are considered and a Q-learning based approach is proposed to estimate the unknown cost distributions. However, no theoretical performance guarantee is provided for this approach.

II. Problem Formulation

In this section, we first introduce the system model for edge caching with uncertain miss costs. Next, we formulate a service cost minimization problem for solving the optimal edge caching policy. We then define the regret to measure the performance of an edge caching policy.

A. System Model

Consider an edge caching scenario with three layers of storage devices composed of edge caches, intermediate caches and backend data centers as illustrated in Fig. 1. Edge caches are placed at the network edges (e.g., on base stations) and are the closest to the end users. Edge caches have very limited cache capacity and can only store a small fraction of the entire dataset. Multiple edge caches are connected to the same intermediate cache. Intermediate caches typically have larger capacity than the edge caches, but still cannot accommodate the entire dataset. Multiple intermediate caches are connected to a backend data center, which stores the entire dataset. In this paper, we focus on a single edge cache and investigate how to develop an optimal content update policy for it.

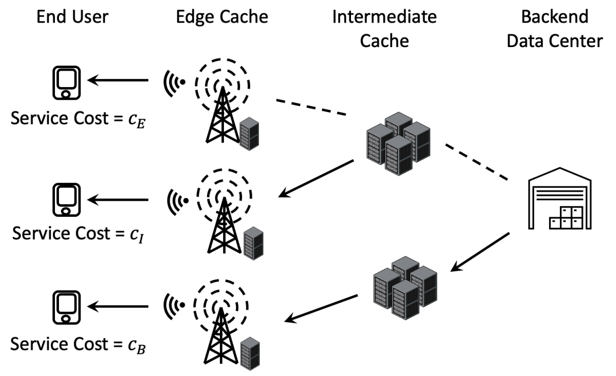

Fig. 1: Edge caching with disparate service costs.

We consider a discrete-time system with time $t = 1, 2, \cdots$. Let $\mathcal{D} = \{d_1, d_2, \cdots, d_N\}$ be a set of N data items, where the item sizes are set to be 1. Assume the capacity of the edge cache is K, $1 \le K < N$. At each time slot t, a data request will be generated and sent to the edge cache. Let R_t denote the data item requested at time t, $R_t \in \mathcal{D}$. We characterize R_t by the Independent Reference Model (IRM), which is commonly assumed in caching analysis [20], [21]. Specifically, the data requests are i.i.d. generated based on the known popularity $p_i \triangleq \mathbb{P}[R_t = d_i]$, with $p_i \in (0, 1)$ and $\sum_{i=1}^{N} p_i = 1$. Once a data request is received, we will first try to serve the request from the edge cache. If the requested data is stored in the edge cache, which is a cache hit, then we can serve the request immediately at a small cost c_E. However, if the data is not stored in the edge cache, (i.e., a cache miss occurs), we need to fetch the data from the other two storage layers and the cost will depend on where we obtain the data. Specifically, we will first try to fetch the data from the intermediate caches. If the requested data is stored there, we can serve the request at a relatively large cost c_I. Otherwise, we have to serve the request from the backend storage which will incur the largest service cost c_B. We assume that c_E, c_I, c_B are known [1] and satisfy $c_B > c_I > c_E \geq 0$. This three-layer storage architecture is commonly used in today's content delivery networks [1].

The key challenge for designing efficient cost-based edge caching policy is that the cost of an edge cache miss depends on whether the missed data is served from the intermediate cache or the backend data centers, which is unknown a priori, because it is impractical for an edge cache to keep track of the data items stored in the intermediate cache. The reason is twofold. First of all, each intermediate cache will serve requests from multiple edge caches. Therefore, it is impossible for an edge cache to infer the data stored in the intermediate cache based on its own traffic. Second, it is impractical for the intermediate cache to notify the edge caches about the data stored in it in real time, since the data stored in the intermediate cache will be updated frequently based on the traffic aggregated from multiple edge caches. Therefore, to model this uncertainty, we use a Bernoulli random variable with unknown parameter to indicate whether a data item is stored in the intermediate cache or not. For each data item d_i, we assume it is *not* stored in the intermediate cache with a probability q_i, $0 < q_i < 1$, independently in each time slot. We assume that q_i is unknown to the edge cache and needs to be estimated for designing efficient edge caching policies. Notably, the design and analysis of this paper could be used in general application scenarios beyond edge caching, as long as the miss costs are modeled by the Bernoulli random variables.

Before we delve into our design and analysis within the above framework, we would like to note that our results can be generalized to allow for unknown item popularities and non-identical item sizes. In particular, the unknown popularities can be integrated by estimating them through observations. Also, incorporating non-identical sizes can be achieved by using approaches similar to those in [2], [6]. Detailed discussions on generalization are presented in the technical report [22].

B. Edge Caching for Cost Minimization

The focus of this paper is to design an efficient edge caching policy that decides which data items should be stored in the edge cache. For each data item d_i and a time horizon n, define

$$T_i^{\text{in}}(n) = \sum_{t=1}^{n} \mathbf{1}(d_i \text{ is stored in the edge cache at time } t),$$

$$T_i^{\text{out}}(n) = \sum_{t=1}^{n} \mathbf{1}(d_i \text{ is not stored in the edge cache at time } t).$$

$T_i^{\text{in}}(n)$ and $T_i^{\text{out}}(n)$ are random variables and depend on the edge caching policy. For a time horizon n, let $Cost(n)$ denote the expected overall service cost accumulated from $t = 1$ to $t = n$. Define

$$\gamma_i = q_i c_B + (1 - q_i) c_I - c_E,$$

[1]Although we know the cost to fetch data items from a specific storage, the miss cost of the edge cache is still uncertain, since the missed item could be fetched from the intermediate cache or the backend data centers, which is unknown a priori.

which can be interpreted as the cost reduction achieved by storing d_i in the edge cache. We have

$$\begin{aligned} Cost(n) =& \sum_{i=1}^{N} \mathbb{E}[T_i^{\text{in}}(n)] p_i c_E \\ &+ \sum_{i=1}^{N} \mathbb{E}[T_i^{\text{out}}(n)] p_i (q_i c_B + (1 - q_i) c_I) \\ =& n c_E + \sum_{i=1}^{N} \mathbb{E}[T_i^{\text{out}}(n)] p_i \gamma_i, \end{aligned} \tag{1}$$

where the last equation holds because $T_i^{\text{in}}(n) + T_i^{\text{out}}(n) = n$ for each data item d_i.

Our objective is to find the edge caching policy that minimizes the expected accumulated cost $Cost(n)$. When the parameter q_i's are known, it is easy to observe that the optimal policy is to always store the data items with the largest $p_i \gamma_i$ values in the edge cache. Without loss of generality, we assume that the data items could be strictly ordered based on $p_i \gamma_i$ and are indexed such that $p_i \gamma_i > p_j \gamma_j$ for any $1 \leq i < j \leq N$. Note that the results of this paper can be easily extended to the case without the strict ordering (i.e., existing $i \neq j$ with $p_i \gamma_i = p_j \gamma_j$). For the sake of better readability and clearer presentation, we focus on the case with strict ordering in this paper. According to the indexing rule, when q_i's are known, the optimal policy stores d_i's, $1 \leq i \leq K$, in the edge cache, where K is the edge cache capacity. We call the set $\{d_i : 1 \leq i \leq K\}$ the optimal choice set, and the set $\{d_i : K + 1 \leq i \leq N\}$ the suboptimal choice set.

In this paper, we consider a more realistic setting where q_i is unknown for reasons that are described in the previous subsection. Consequently, γ_i is also unknown, being a function of q_i. Thus, we need to adaptively learn q_i's and update the caching content accordingly. In particular, a caching policy needs to make the following decisions. When an edge cache miss occurs, we will fetch the requested data item from the intermediate cache or, if not in the intermediate cache, from the backend data center, and decide whether to load it into the edge cache. And if the edge cache is full, we need to decide which data item should be evicted to accommodate this new item. Notably, we are not allowed to load a data item into the edge cache if it is not requested, because this will incur additional costs to fetch it.

The performance of edge caching policies will be evaluated by regrets. This is a classical learning metric which characterizes the difference between the expected accumulated cost achieved by an edge caching policy and the one achieved by the optimal policy with known q_i's. Formally, we define the regret over a time horizon n as

$$\begin{aligned} Regret(n) =& Cost(n) \\ &- n \left(\sum_{i=1}^{K} p_i c_E + \sum_{i=K+1}^{N} p_i (q_i c_B + (1 - q_i) c_I) \right). \end{aligned}$$

Minimizing the expected accumulated cost $Cost(n)$ is equivalent to minimizing $Regret(n)$.

III. Motivation and Challenges

In this section, we first introduce our motivation by showing that a natural heuristic design could achieve significant inefficiency. To solve this issue, we propose to leverage online learning to adaptively estimate the unknown parameters. However, existing online learning algorithms do not consider the specific constraints and dynamics in caching systems and therefore cannot be applied directly. We then introduce the challenges of combining online learning and caching.

A. Motivation: Substandard Performance of Heuristic Designs

An edge caching policy needs to estimate the unknown parameter q_i based on history information and make caching decisions accordingly. Define

$$T_i^{\text{miss}}(t) = \sum_{s=1}^{t} \mathbf{1}(d_i \text{is requested and missed from the edge cache at time } s), \tag{2}$$

$$T_i^{\text{back}}(t) = \sum_{s=1}^{t} \mathbf{1}(d_i \text{is requested and served from the backend data center at time } s). \tag{3}$$

Recall that q_i is the probability that the data item d_i is not stored in the intermediate cache. An unbiased estimation of q_i at time t is the sample mean

$$\hat{q}_i(t) = T_i^{\text{back}}(t)/T_i^{\text{miss}}(t). \tag{4}$$

A heuristic caching policy is to estimate γ_i by

$$\hat{\gamma}_i(t) = \hat{q}_i(t)c_B + (1 - \hat{q}_i(t))\, c_I - c_E, \tag{5}$$

and evict the data item with the smallest $p_i\hat{\gamma}_i(t)$ value when the cache is full. We formally describe this heuristic edge caching policy in Algorithm 1. Next, we will introduce a toy example showing that the heuristic policy could be problematic.

Algorithm 1: Heuristic Edge Caching Policy

```
1  Initialization: $T_i^{\text{miss}}(0) = T_i^{\text{back}}(0) = \hat{q}_i(0) = 0$,
     $\hat{\gamma}_i(0) = c_I - c_E$, $1 \le i \le N$;
2  for $t = 1 : n$ do
3      if $R_t$ is not stored in the edge cache then
4          Let $i$ denote the index of $R_t$;
5          Update $T_i^{\text{miss}}(t)$,$T_i^{\text{back}}(t)$, $\hat{q}_i(t)$ and $\hat{\gamma}_i(t)$ based
             on (2), (3), (4) and (5), respectively;
6          if Edge cache is full then
7              $j = argmin\{p_k\hat{\gamma}_k(t) :$
                 $d_k$ is currently stored in the edge cache$\}$;
8              if $p_i\hat{\gamma}_i(t) > p_j\hat{\gamma}_j(t)$ then
9                  Load $d_i$ into the edge cache and evict
                     $d_j$;
10 end
```

Example 1. *Consider two data items and an edge cache with size* 1 *(i.e., $K = 1$). Set $p_1 = p_2 = 0.5$, $q_1 = 0.5$, $q_2 = 0.1$,*

Time t	1	2	3	4	5
Data request R_t	d_2	d_1	d_2	d_1	d_2
d_1 in intermediate cache	True	True	False	False	True
d_2 in intermediate cache	False	True	True	True	True
$\hat{q}_1(t)$	0	0	0	1/2	1/2
$\hat{q}_2(t)$	1	1	1	1	1
Edge cache content	d_2	d_2	d_2	d_2	d_2

TABLE I: Request trace for the toy example.

$c_E = 1, c_I = 2, c_B = 10$. Table I gives an example trace for data requests as well as whether the requested data is stored in the intermediate cache. Assume that the edge cache is initially empty. We calculate $\hat{q}_i(t)$ and make caching decisions based on the heuristic policy. Note that the optimal solution is to store d_1 in the edge cache. However, for this example, d_1 will never be cached by the heuristic edge caching policy.

- *For $t = 1$, d_2 is requested. Since the edge cache is initially empty, it is a miss. Moreover, d_2 is not stored in the intermediate cache. We will fetch d_2 from the backend data center and load it into the edge cache. According to (4), q_2 is estimated as 1 at $t = 1$.*
- *For $t = 2$, d_1 is requested and missed. We fetch it from the intermediate cache and update the estimation for q_1. Since $p_2\hat{\gamma}_2(2) > p_1\hat{\gamma}_1(2)$, we will not update the cache content.*
- *For t ≥ 3, since the request for d_2 can be directly served from the edge cache, we will not be able to observe whether d_2 is stored in the intermediate cache and update $\hat{q}_2(t)$. Consequently, $p_1\hat{\gamma}_1(t) < p_2\hat{\gamma}_2(t)$ always holds. The heuristic policy will always store d_2 in the edge cache regardless of future data requests, and achieves substandard performance.*

This example reveals a severe problem of the heuristic edge caching policy: the inaccurate estimation at the early stage could make the edge cache stop collecting observations for already cached data items and get stuck in a suboptimal solution. In order to solve this issue, we are motivated to leverage online learning techniques to estimate q_i and update cache content strategically.

B. Our Approach: Adaptive Caching via Online Learning

We first highlight an exploration and exploitation tradeoff for the proposed edge caching problem:
Exploration: On the one hand, we would like to not store data items in the edge cache intentionally, since this could trigger cache misses and collect more observations on the miss cost, and therefore, help us make more accurate estimation for q_i's.
Exploitation: On the other hand, we want to exploit the current estimation and cache the items with the largest $p_i\hat{\gamma}_i(t)$. This could potentially minimize the overall service costs.

The proposed heuristic policy only exploits but never explores, and therefore performs quite poorly. In order to efficiently learn the unknown parameters and meanwhile achieve good caching performance, we have to balance the exploration and exploitation tradeoff. This tradeoff has been extensively studied in online learning literature (in particular, the multi-armed bandit (MAB) problems [7]). The key idea of MAB algorithms is to encourage exploration by adding an underestimate of q_i that changes with time and new samples. As a result, a cached data item will be eventually evicted when the estimated q_i is small enough. Then, a new observation for the miss cost will be observed, and the estimation will be updated. However, the conventional algorithm and analysis for MAB problems cannot be directly applied to the edge caching system due to the additional cache capacity and cache update constraints, which makes the problem highly non-trivial.

C. Key Challenge: Learning with Caching Constraints

First, we point out that the proposed edge caching problem share some similarities with conventional combinatorial multi-armed bandit problem. In particular, it is similar to stochastic combinatorial semi bandits with N arms in total and $N-K$ arms played at each time slot. Once an arm is played, a random cost will be incurred. The cost of playing the i^{th} arm is $p_i(c_B - c_E)$ with probability q_i and $p_i(c_I - c_E)$ with probability $1-q_i$. And the total cost of each time slot is equal to the sum of the costs incurred by the $N-K$ played arms. The objective of this bandit problem is to minimize the overall accumulated costs.

Notably, the edge caching problem has additional constraints compared with the combinatorial bandit problem:

- At each time slot, the edge caching policy can only update one cached data item, while the bandit problem has no such constraints.
- The data items that can be stored at time t depend on an external random process, (i.e., the data request), as well as the cache content at the previous time slot $t-1$. In contrast, the action set of bandit problems typically does not depend on time or external random processes.

These constraints in edge caching systems make the problem more challenging. Consequently, the standard bandit algorithms and analysis cannot be directly applied. To solve this issue, we need to propose novel edge caching policies and use new theoretical tools to analyze its performance. To that end, in the next section, we derive a regret lower bound over all policies of interest, followed in the subsequent section by a new design with a novel regret upper bound that matches the scaling of the lower bound, thereby proving its asymptotic-optimality.

IV. Regret Lower Bound

Before designing edge caching polices, we first derive a lower bound for the regret performance of all "good" policies. Following the approach of the seminal work [23], we say that an edge caching policy is a uniformly good policy, if the regret achieved by it satisfies $Regret(n) = o(n^{\alpha})$ for $\forall \alpha > 0$. Let $D_{KL}(p,q)$ denote the Kullback–Leibler divergence for two Bernoulli random variables with parameter p and q, respectively. We have for $0 < p < 1$ and $0 < q < 1$,

$$D_{KL}(p,q) = p \log \frac{p}{q} + (1-p) \log \frac{1-p}{1-q}.$$

We prove a regret lower bound in the following theorem.

Theorem 1. *For any uniformly good policy, the regret satisfies*

$$\liminf_{n \to +\infty} \frac{Regret(n)}{\log n} \geq \sum_{i \in \mathcal{S}} \frac{p_i\gamma_i - p_{K+1}\gamma_{K+1}}{D_{KL}\left(q_i, \frac{p_{K+1}\gamma_{K+1} - p_i(c_I - c_E)}{p_i(c_B - c_I)}\right) \cdot p_i},$$

where $\mathcal{S} = \{1 \leq i \leq K : p_i(c_I - c_E) < p_{K+1}\gamma_{K+1}\}$.

Theorem 1 shows that the regret increases at least logarithmically with the particular coefficient on the right hand side asymptotically with the time horizon n, if the set $\mathcal{S}$ is not empty. Recall that the data items are indexed such that $p_i\gamma_i$ is decreasing. We make a few remarks for this result:

- The constant on the right hand side of the regret lower bound depends on the "distance", as measured by the $p_i\gamma_i$ values and a specific form of the Kullback–Leibler divergence, between the data items in the optimal choice set (i.e, $d_i, 1 \leq i \leq K$) and the best data item in the suboptimal choice set (i.e., d_{K+1}).
- The scaling is independent of the total number of items N, i.e., arms, in the dataset $\mathcal{D}$. This is different from most MAB regret performances where the number of arms is a scaling factor. This property arises due to the particular structure of our caching system, whereby subsets of data items (i.e., arms) are being selected.
- The proof of this theorem is not a simple application of the proof for classic MAB problems, considering the data request dynamics and cache capacity and update constraints of the edge caching systems. Detailed proofs are presented in the technical report [22].

Moreover, note that for $d_i \notin \mathcal{S}$ with $1 \leq i \leq K$ and d_j with $K+1 \leq j \leq N$, we must have

$$p_i\gamma_i = p_i(q_i c_B + (1-q_i)c_I - c_E) > p_i(c_I - c_E) \geq p_{K+1}\gamma_{K+1} \geq p_j\gamma_j,$$

for $\forall q_i \in (0,1)$, which indicates that we could easily distinguish d_i from the suboptimal choice set (i.e., d_j, $K+1 \leq j \leq N$), even when the estimation of q_i is arbitrarily chosen. Thus, the scenario with empty $\mathcal{S}$ degenerates to a trivial problem and is not the main focus of this paper.

V. KL-LCB Based Edge Caching Policy

In this section, we first propose a novel edge caching policy that leverages online learning ideas. Then, we prove that the proposed policy achieves asymptotically optimal regret. Instead of estimating q_i by the sample mean $\hat{q}_i(t)$ like the heuristic policy, we follow the principle of "optimism in the face of uncertainty" [7] and use an underestimate $\tilde{q}_i(t)$, which is defined as

$$\tilde{q}_i(t) = \min\left\{q \in (0,1) : D_{KL}(\hat{q}_i(t), q) \leq \frac{\log f(t)}{T_i^{\text{miss}}(t)}\right\}, \quad (6)$$

where $\hat{q}_i(t)$ is defined in (4) and $f(t) = 1 + t(\log t)^2$. We have $0 < \tilde{q}_i(t) \leq \hat{q}_i(t)$ and the "distance" between $\tilde{q}_i(t)$ and $\hat{q}_i(t)$ is characterized by $\log f(t)/T_i^{\text{miss}}(t)$. This design is inspired by the KL-UCB algorithm for reward maximization in conventional stochastic MAB problems [24], [25]. For the edge caching problem, our objective is cost minimization rather than reward maximization. Therefore, we apply the KL-LCB based design, which is symmetric to KL-UCB.

Next, we estimate γ_i by

$$\tilde{\gamma}_i(t) = \tilde{q}_i(t)c_B + (1 - \tilde{q}_i(t))\, c_I - c_E. \tag{7}$$

Based on $\tilde{\gamma}_i(t)$, we propose a KL-LCB based edge caching policy, which is described in Algorithm 2. When the data request is served by the edge cache, we do not update the cache content. When the requested data (e.g., d_i) is not stored in the edge cache, we will fetch d_i from the intermediate cache, or, if it is not stored there, from the backend data center. Then, we find the cached data item d_j with the smallest $p_j\tilde{\gamma}_j(t)$ values among all cached data items. If $p_i\tilde{\gamma}_i(t) > p_j\tilde{\gamma}_j(t)$, then we will replace the cached data item d_j by the newly-requested data item d_i. On the one hand, the proposed policy attempts to cache the data items with large $p_i\tilde{\gamma}_i(t)$ values, which exploits the current knowledge. On the other hand, $\log(t)/T_i^{\text{miss}}(t)$ will increase with time t, if there are no misses observed. Consequently, if d_i is currently cached, $\tilde{q}_i(t)$ as well as $\tilde{\gamma}_i(t)$ will gradually decrease with time t. And d_i will finally be evicted when $p_i\tilde{\gamma}_i(t)$ is small enough, which encourages exploration.

Algorithm 2: KL-LCB based Edge Caching Policy

```
1  Initialization: $T_i^{\text{miss}}(0) = T_i^{\text{back}}(0) = \hat{q}_i(0) = \tilde{q}_i(0) = 0$,
     $\hat{\gamma}_i(0) = c_I - c_E$, $1 \le i \le N$;
2  for $t = 1 : n$ do
3      if $R_t$ is not stored in the edge cache then
4          Let $i$ denote the index of $R_t$;
5          Update $T_i^{\text{miss}}(t)$, $T_i^{\text{back}}(t)$, $\hat{q}_i(t)$, $\tilde{q}_i(t)$ and $\tilde{\gamma}_i(t)$
             based on (2), (3), (4), (6) and (7), respectively;
6          if Edge cache is full then
7              $j = argmin\{p_k\tilde{\gamma}_k(t)$ :
                 $d_k$ is currently stored in the edge cache$\}$;
8              if $p_i\tilde{\gamma}_i(t) > p_j\tilde{\gamma}_j(t)$ then
9                  Load $d_i$ into the edge cache and evict
                     $d_j$;
10 end
```

A. Regret Upper Bound and Asymptotic Optimality

We first provide theoretical performance guarantees for the proposed KL-LCB based edge caching policy by deriving a regret upper bound in Theorem 2.

Theorem 2. *For the proposed KL-LCB based policy, we have*

$$\limsup_{n\to+\infty} \frac{Regret(n)}{\log n} \le \sum_{i\in\mathcal{S}} \frac{p_i\gamma_i - p_{K+1}\gamma_{K+1}}{D_{KL}\left(q_i, \frac{p_{K+1}\gamma_{K+1} - p_i(c_I - c_E)}{p_i(c_B - c_I)}\right)\cdot p_i},$$

where $\mathcal{S} = \{1 \le i \le K : p_i(c_I - c_E) < p_{K+1}\gamma_{K+1}\}$.

It is easy to observe that the upper bound derived in Theorem 2 matches the lower bound derived in Theorem 1. Thus, we can conclude that the KL-LCB based edge caching policy is *asymptotically optimal* if the set $\mathcal{S}$ is not empty.

It is well known that the KL-UCB policy achieves asymptotically optimal regrets for conventional stochastic MAB problems with Bernoulli-distributed rewards. One question is whether the KL-LCB based edge caching policy proposed in this paper is a simple application of the KL-UCB policy. The answer is no. Although the miss cost estimation in the proposed edge caching policy is symmetric to the reward estimation in KL-UCB, the theoretical analysis of the proposed edge caching policy is much more challenging for the following reasons:

- The nature of edge caching systems imposes significant complexity into performance analysis. Specifically, the caching capacity constraint introduces a non-traditional combinatorial structure, i.e., multiple data items could be cached at each time slots. The cache update rule brings time correlations into action sets. In particular, the data items that can be cached at a time slot t depend on the data items that were already cached at the previous time slot $t-1$, while in the original KL-UCB setting, the action sets have no such time correlations.
- The data request process introduces new dynamics compared to the original MAB settings. Specifically, the caching decisions (i.e., the action set) at each slot depend on a random data request, while in the original KL-UCB setting, there is no such dependency.

Simply applying existing analysis of the KL-UCB policy cannot resolve these challenges. Instead, we need to develop novel approaches in this paper to analyze the theoretical performance. In particular,

- We characterize the nature of the caching systems by exploring the relationship between $T_i^{\text{miss}}(n)$, $T_i^{\text{out}}(n)$ and $T_i^{\text{in}}(n)$, $1 \le i \le N$.
- Instead of showing the convergence in expectation for critical statistics as in the conventional MAB analysis, our setting requires a much stronger almost-sure convergence results, which require non-trivial new analysis.

Next, we will show a sketch of proof for the key Theorem 2.

B. Sketch of Proof for Theorem 2

Due to the page limit, we present the proof of Theorem 2 in the technical report [22] and introduce a sketch of the proof in this section. We start by establishing a few lemmas.

Lemma 1. *The regret of the KL-LCB based edge caching policy can be upper bounded as*

$$Regret(n) \le \sum_{i=1}^{K} \mathbb{E}[T_i^{out}(n)]p_i\gamma_i - \mathbb{E}[T_{K+1}^{in}(n)]p_{K+1}\gamma_{K+1}.$$

Lemma 1 provides a regret upper bound related to the costs introduced by the optimal choice set (i.e., $d_i, 1 \le i \le K$) and best suboptimal choice (i.e., d_{K+1}). To connect this upper bound with Theorem 2, we need to characterize $\mathbb{E}[T_i^{\text{out}}(n)]$, $1 \leq i \leq K$, and $\mathbb{E}[T_{K+1}^{\text{in}}(n)]$ under the proposed KL-LCB based policy.

Lemma 2. *Under the KL-LCB based policy, we have, for $1 \leq i \leq K$, if $p_i(c_I - c_E) < p_{K+1}\gamma_{K+1}$,*

$$\limsup_{n\to+\infty} \frac{\mathbb{E}\left[T_i^{out}(n)\right]}{\log n} \leq 1\bigg/\left(p_i \cdot D_{KL}\left(q_i, \frac{p_{K+1}\gamma_{K+1} - p_i(c_I - c_E)}{p_i(c_B - c_I)}\right)\right), \quad (8)$$

and $\lim_{n\to+\infty} \mathbb{E}\left[T_i^{out}(n)\right]/\log n = 0$ *if* $p_i(c_I - c_E) \geq p_{K+1}\gamma_{K+1}$

Lemma 2 shows an upper bound for $\mathbb{E}[T_i^{\text{out}}(n)]$, $1 \leq i \leq K$, under the proposed KL-LCB based policy. Next, we will prove a relationship between $\mathbb{E}[T_i^{\text{out}}(n)]$, $1 \leq i \leq K$, and $\mathbb{E}[T_{K+1}^{\text{in}}(n)]$, which is the most critical and challenging part for the proof of Theorem 2.

Lemma 3. *If $p_i(c_I - c_E) < p_{K+1}\gamma_{K+1}$ for some $1 \leq i \leq K$, then under the KL-LCB based edge caching policy, we have*

$$\lim_{n\to+\infty} \frac{\mathbb{E}[T_{K+1}^{in}(n)]}{\sum_{i=1}^K \mathbb{E}[T_i^{out}(n)]} = 1.$$

Lemma 3 indicates that under the KL-LCB based edge caching policy, the duration of time when the cache content is not the optimal choice set (i.e., $\sum_{i=1}^K \mathbb{E}[T_i^{\text{out}}(n)]$) is asymptotically equal to the duration of time when the best suboptimal choice is stored in the cache (i.e, $\mathbb{E}[T_{K+1}^{\text{in}}(n)]$). In other words, when the time horizon n is large, the proposed edge caching policy only gets confused with the best suboptimal data item d_{K+1}, and is pretty sure that other suboptimal data items (i.e., d_i, $K+2 \leq i \leq N$) should not be cached. Intuitively, this result makes a lot of sense, because other suboptimal data items have a larger gap with the optimal choice set and should be easier to distinguish. However, proving this fact rigorously requires establishing the almost sure convergence for critical statistics including $\tilde{q}_i(t)$, $T_i^{\text{miss}}(t)$, etc.

With these established lemmas, we are ready to prove Theorem 2.

Proof of Theorem 2. Combining Lemmas 1, 2 and 3, we have

$$\limsup_{n\to+\infty} \frac{Regret(n)}{\log n} \leq \limsup_{n\to+\infty} \frac{\sum_{i=1}^K \mathbb{E}[T_i^{\text{out}}(n)]p_i\gamma_i - \mathbb{E}[T_{K+1}^{\text{in}}(n)]p_{K+1}\gamma_{K+1}}{\log n} \quad (9)$$

$$= \limsup_{n\to+\infty} \frac{\sum_{i=1}^K \mathbb{E}[T_i^{\text{out}}(n)](p_i\gamma_i - p_{K+1}\gamma_{K+1})}{\log n} \quad (10)$$

$$\leq \sum_{i\in\mathcal{S}} \frac{p_i\gamma_i - p_{K+1}\gamma_{K+1}}{D_{KL}\left(q_i, \frac{p_{K+1}\gamma_{K+1} - p_i(c_I - c_E)}{p_i(c_B - c_I)}\right) \cdot p_i}, \quad (11)$$

where $\mathcal{S} = \{1 \leq i \leq K : p_i(c_I - c_E) < p_{K+1}\gamma_{K+1}\}$. Note that Equations (9), (10) and (11) leverage Lemmas 1, 3 and 2, respectively. ☐

VI. Evaluation

In this section, we will evaluate the empirical performance of the proposed policies. In Experiment 1, we evaluate the regret achieved by the KL-LCB based policy as well as the heuristic policy. In Experiment 2, we consider the scenario where the data popularities are unknown. We simulate a generalized KL-LCB based policy and compare it with a few benchmarks including LRU, LFU and a generalized heuristic policy that is a variant of GDSF.

Experiment 1: In this experiment, we evaluate the performance achieved by the proposed KL-LCB based edge caching policy (i.e., Algorithm 2) and the heuristic policy (i.e., Algorithm 1). Consider a total number of 1000 data items. Set the cache capacity $K = 100$, $c_E = 1$, $c_I = 5$, $c_B = 100$. Assume that data popularities follow a Zipf's distribution, which has been validated by real data traces [26], [27], [1]. Specifically, set $p_i = c \cdot i^{-0.4}$, $1 \leq i \leq 1000$, where $c = 1/\sum_{i=1}^{1000} i^{-0.4} = 9.61 \times 10^{-3}$ is the normalization factor. Set $q_i = 0.2$ for $1 \leq i \leq 500$ and $q_i = 0.9$ for $501 \leq i \leq 1000$. Thus, a more popular data item will have a larger probability (i.e., $1 - q_i$) to be stored in the intermediate cache. Notably, in previous sections, we assumed that data items are indexed such that $p_i\gamma_i$ is decreasing for the ease of analysis and presentation. However, in this experiment, the data items are indexed such that the popularity p_i is decreasing.

We simulate the proposed KL-LCB based policy and the heuristic policy for a time horizon n ranging from 2000 to 20000. Each experiment is repeated for 30 times to obtain the average accumulated costs. Regrets are calculated based on the average accumulated costs and plotted in Fig. 2. It can be observed that the KL-LCB based policy achieves much better performance than the heuristic policy. Moreover, the heuristic policy achieves a linear regret, while the KL-LCB policy achieves a sublinear (theoretically proven to be logarithmic) regret, which has been established in Theorems 1 and 2.

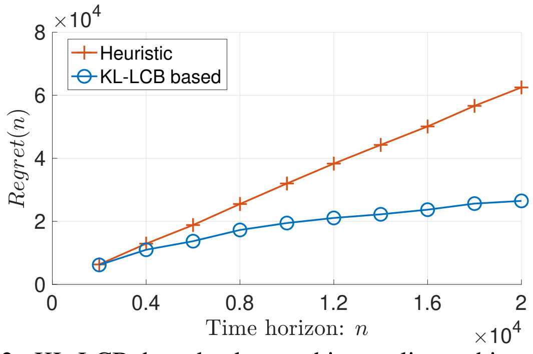

Fig. 2: KL-LCB based edge caching policy achieves better performance than the heuristic policy.

Experiment 2: In this experiment, we consider the scenario where data popularities are unknown. To handle unknown popularities, in each time slot t, we estimate p_i by $\hat{p}_i(t) = \sum_{s=1}^t \mathbf{1}(d_i \text{ is requested at time } s)/t$, and generalize the heuristic and the KL-LCB based edge caching policies by replacing p_i in Algorithms 1 and 2 with $\hat{p}_i(t)$. More details of this generalization are presented in the technical report [22]. Note that the generalized heuristic policy is a variant of the GDSF policy [2] for unknown costs and identical data sizes. We also simulate the LFU and LRU polices, which are commonly adopted for unknown popularity, but do not account for the service costs. We use the same parameter setting as in Experiment 1, and present the numerical results in Fig. 3. We can observe that the KL-LCB based policy achieves

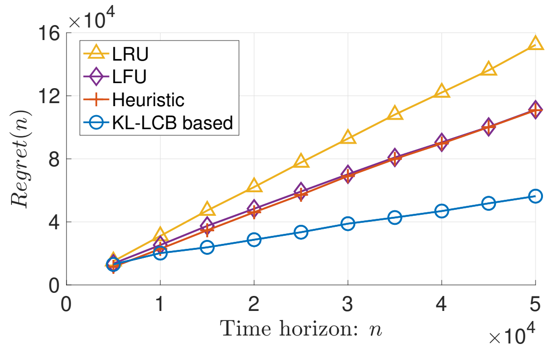

Fig. 3: KL-LCB based edge caching policy generalized for unknown data popularity.

distinctly smaller regrets than all the alternatives. Interestingly, the heuristic policy achieves similar performance with LFU that does not employ any cost information. This insightful observation indicates that the heuristic policy cannot fully utilize the cost information and validates the importance of the carefully designed cost learning in the KL-LCB based policy. The LRU policy achieves the worst performance, since it does not fully utilize historical request records or cost information.

VII. Conclusion

Existing cost-based caching policies typically assume known miss costs, which is not always the case in real systems. To address this issue, we focused on an edge caching scenario with unknown miss costs and developed novel caching policies that learn the miss costs efficiently. By presenting a carefully-designed example, we first show that a heuristic learning design could induce significant caching inefficiency. We then derived a regret lower bound for any uniformly good policy. Inspired by the "optimism in the face of uncertainty" principle in online learning literature, we developed a KL-LCB based edge caching policy and proved that it achieves the regret lower bound and therefore is asymptotically optimal. Extensive numerical experiments indicate the proposed policy significantly improves caching performance over other benchmarks. The novel techniques used in this work to handle caching constraints and dynamics could be potentially leveraged to design and analyze learning mechanisms for other systems.

References

[1] Q. Huang, K. Birman, R. Van Renesse, W. Lloyd, S. Kumar, and H. C. Li, "An analysis of facebook photo caching," in *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*, 2013, pp. 167–181.

[2] L. Cherkasova, *Improving WWW proxies performance with greedy-dual-size-frequency caching policy*. Hewlett-Packard Laboratories, 1998.

[3] P. Cao and S. Irani, "Cost-aware WWW proxy caching algorithms." in *Usenix symposium on internet technologies and systems*, vol. 12, no. 97, 1997, pp. 193–206.

[4] C. Li and A. L. Cox, "Gd-wheel: a cost-aware replacement policy for key-value stores," in *Proceedings of the Tenth European Conference on Computer Systems*, 2015, pp. 1–15.

[5] N. Young, *Competitive paging and dual-guided on-line weighted caching and matching algorithms*. Princeton University, 1991.

[6] A. Blankstein, S. Sen, and M. J. Freedman, "Hyperbolic caching: Flexible caching for web applications," in *2017 USENIX Annual Technical Conference (USENIX ATC 17)*, 2017, pp. 499–511.

[7] T. Lattimore and C. Szepesvári, *Bandit algorithms*. Cambridge University Press, 2020.

[8] I. Szita and A. Lőrincz, "The many faces of optimism: a unifying approach," in *Proceedings of the 25th international conference on Machine learning*, 2008, pp. 1048–1055.

[9] P. Cao and S. Irani, "Greedydual-size: A cost-aware www proxy caching algorithm," in *2nd Web Caching Workshop, Boulder, Colorado*, 1997.

[10] B. Hou and F. Chen, "Gds-lc: A latency-and cost-aware client caching scheme for cloud storage," *ACM Transactions on Storage (TOS)*, vol. 13, no. 4, pp. 1–33, 2017.

[11] J. Shim, P. Scheuermann, and R. Vingralek, "Proxy cache algorithms: Design, implementation, and performance," *IEEE Transactions on Knowledge and Data Engineering*, vol. 11, no. 4, pp. 549–562, 1999.

[12] S. Liang, K. Chen, S. Jiang, and X. Zhang, "Cost-aware caching algorithms for distributed storage servers," in *International Symposium on Distributed Computing*. Springer, 2007, pp. 373–387.

[13] J. Song, M. Sheng, T. Q. Quek, C. Xu, and X. Wang, "Learning-based content caching and sharing for wireless networks," *IEEE Transactions on Communications*, vol. 65, no. 10, pp. 4309–4324, 2017.

[14] A. Sengupta, S. Amuru, R. Tandon, R. M. Buehrer, and T. C. Clancy, "Learning distributed caching strategies in small cell networks," in *2014 11th International Symposium on Wireless Communications Systems (ISWCS)*. IEEE, 2014, pp. 917–921.

[15] X. Xu, M. Tao, and C. Shen, "Collaborative multi-agent multi-armed bandit learning for small-cell caching," *IEEE Transactions on Wireless Communications*, vol. 19, no. 4, pp. 2570–2585, 2020.

[16] A. Bura, D. Rengarajan, D. Kalathil, S. Shakkottai, and J.-F. Chamberland, "Learning to cache and caching to learn: Regret analysis of caching algorithms," *IEEE/ACM Transactions on Networking*, 2021.

[17] P. Blasco and D. Gündüz, "Multi-armed bandit optimization of cache content in wireless infostation networks," in *2014 IEEE International Symposium on Information Theory*. IEEE, 2014, pp. 51–55.

[18] ——, "Learning-based optimization of cache content in a small cell base station," in *2014 IEEE International Conference on Communications (ICC)*. IEEE, 2014, pp. 1897–1903.

[19] A. Sadeghi, F. Sheikholeslami, A. G. Marques, and G. B. Giannakis, "Reinforcement learning for adaptive caching with dynamic storage pricing," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 10, pp. 2267–2281, 2019.

[20] S. Vanichpun and A. M. Makowski, "The output of a cache under the independent reference model: where did the locality of reference go?" in *Proceedings of the joint international conference on Measurement and modeling of computer systems*, 2004, pp. 295–306.

[21] R. Fagin and T. G. Price, "Efficient calculation of expected miss ratios in the independent reference model," *SIAM Journal on Computing*, vol. 7, no. 3, pp. 288–297, 1978.

[22] G. Quan, A. Eryilmaz, and N. Shroff. (2022) Regret-optimal learning for minimizing edge caching service costs (technical report). [Online]. Available: https://www2.ece.ohio-state.edu/~eryilmaz/learningCaching-TR.pdf

[23] T. L. Lai and H. Robbins, "Asymptotically efficient adaptive allocation rules," *Advances in applied mathematics*, vol. 6, no. 1, pp. 4–22, 1985.

[24] T. L. Lai, "Adaptive treatment allocation and the multi-armed bandit problem," *The Annals of Statistics*, pp. 1091–1114, 1987.

[25] O. Cappé, A. Garivier, O.-A. Maillard, R. Munos, and G. Stoltz, "Kullback-leibler upper confidence bounds for optimal sequential allocation," *The Annals of Statistics*, pp. 1516–1541, 2013.

[26] Y. Yang and J. Zhu, "Write skew and Zipf distribution: Evidence and implications," *ACM transactions on Storage (TOS)*, vol. 12, no. 4, pp. 1–19, 2016.

[27] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web caching and zipf-like distributions: Evidence and implications," in *IEEE INFOCOM'99. Conference on Computer Communications. Proceedings. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. The Future is Now (Cat. No. 99CH36320)*, vol. 1. IEEE, 1999, pp. 126–134.