On the Regret of Online Coded Caching

Anonymous authors

***Abstract*—We consider the widely studied problem of 'coded caching under non-uniform requests' where users independently request files according to some underlying popularity distribution in each slot. This work is a first step towards analyzing this framework through the lens of online learning. We consider the case where the underlying request distribution is apriori unknown and propose an online policy as well as study its regret with respect to an oracle which knows the underlying distribution and employs a well-known order-optimal placement and coded delivery strategy. We also bound the switching cost of this strategy and also discuss a lower bound on the regret of any online scheme in a restricted but natural class of policies.**

I. Introduction

The last few decades have witnessed an unprecedented growth of demand for high-definition content over the internet. The resulting increased load on the underlying communication networks has been mitigated by the widespread adoption of content delivery networks, which deploy storage devices or caches across large geographical regions. These caches are then used to pre-fetch popular content in the off-peak hours. This cached content can then be used to reduce network traffic during peak hours when users make the most requests.

Caching has a rich history, see for example [1] and references therein. More recently, an information-theoretic study of caching networks began with the seminal work of Maddah-Ali and Niesen [2], [3]. This has led to a large amount of literature, under the moniker 'coded caching', studying various aspects of such systems, including non-uniform content popularity [4]–[8], network topology [5], [9], security and privacy [10], [11] etc. Broadly speaking, these works propose content placement and delivery schemes and then compare their performance with that of the information-theoretic optimal, often showing a gap of at most a constant multiplicative factor independent of the system size.

In particular, our work considers the framework of 'coded caching under non-uniform requests' [4], [5], [12] where at each time, users request files according to an underlying request distribution. The aforementioned works assume that the request distribution is known and then design order-optimal placement and delivery schemes. In this work, we consider a scenario where the request distribution is apriori unknown, and the system has to adapt its content placement and delivery scheme over time based solely on the requests observed at the users. We measure the performance of any such *online policy* using the metric of *regret*, which is standard in the online learning literature. For any given time horizon T, the regret of a scheme considers the cumulative additive gap with respect to an oracle which knows the underlying distribution beforehand. We propose a natural online policy for the coded caching framework and demonstrate that its regret is bounded by an instance-dependent constant, and thus does not scale with the time horizon. Given that the online policy changes the content placement across time, we also consider the switching cost of our policy in terms of the number of timesteps where the cached content needs to be updated and are again able to show a constant upper bound for this cost. Finally, we also prove a lower bound on the regret of any online policy in a certain restricted class.

The study of caching systems in the framework of online learning has been pursued recently by other works as well. Most of these focus on the case of a single cache. Inspired by the recent advances in online convex optimization [13]–[15], several online caching policies have been proposed including Online Gradient Ascent [16], [17], Online Mirror Descent [18], and Follow the Perturbed Leader (FTPL) [19]–[21]. These works consider the case of *adversarial requests* and demonstrate that an order-optimal regret of $\Theta(\sqrt{T})$ can be achieved. On the other hand, there have also been some works which consider *stochastic requests* [21], [22] and establish tighter bounds on the expected regret. Our work is similar in spirit and is a first step to studying the popular coded caching framework from the lens of online learning.

The rest of the paper is organized as follows. We describe the problem setup formally in Section II. In Section III, we describe our proposed online caching policy and establish upper bounds on its regret as well as switching cost. Section IV considers a restricted but natural class of online policies and derives a lower bound on the regret of any policy in this class. Numerical results are presented in section V. We conclude with a short discussion in Section VI. Due to lack of space, the proofs of some of the theorems and lemmas stated are presented in [23].

II. System Model

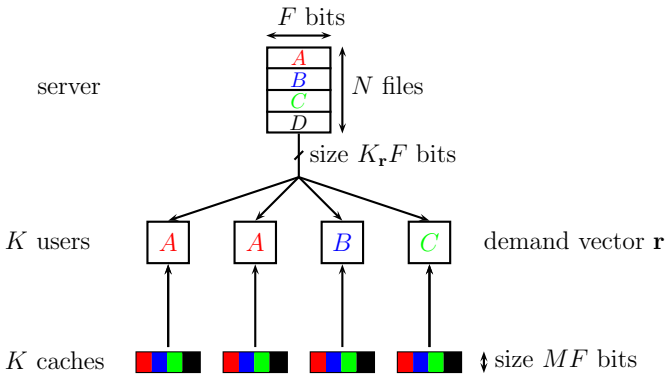


Fig. 1. System setup for the coded caching problem

Consider the caching system depicted in Figure 1. It consists of a single server hosting N files $W_1, W_2, \ldots, W_N$, each of size F bits. The server is connected via an error-free broadcast link to K users, each equipped with a local cache of size MF bits. Consider time to be divided into slots and in each slot t, the system operates in two phases: a *placement* phase followed by a *delivery* phase. The placement phase happens during the low network traffic hours, and in this phase, the cache at each user k is populated with content related to the N files, say $Z_k(t)$. Note that the content stored in each cache can vary over time, possibly as a function of prior requests. Next, during the delivery phase, each user k requests the file with index $r_k^t \in \{1, 2, \ldots, N\}$ which is generated according to an underlying probability mass function (pmf) $\mathbf{p} = (p_1, p_2, \ldots, p_N)$. Without loss of generality, we assume $p_1 \geq p_2 \geq \cdots \geq p_N$. The requests are assumed to be independent and identically distributed (i.i.d.) across time and users. Furthermore, the request distribution $\mathbf{p}$ is apriori unknown to the caching system. The request profile $\mathbf{r^t} = (r_1^t, r_2^t, \ldots, r_K^t)$ is forwarded to the server, and depending on the requests and the cached contents, the server transmits a message $X(t)$ of size $K_{\mathbf{r^t}} F$ bits on the shared link to the users. Each user k should be able to recover its requested file $W_{r_k^t}$ using $X(t)$ and its cached content $Z_k(t)$. For any feasible online (placement + delivery) policy π and over any given time horizon T, the overall (normalized) expected server transmission size is given by $\mathcal{K}^\pi(T) = \mathbb{E}_{\mathbf{p}}[\sum_{t=1}^T K_{\mathbf{r^t}}]$, where the expectation is with respect to the underlying request distribution $\mathbf{p}$.

In the same spirit as standard online learning problems, we will compare the performance of any online policy with that of a (static) oracle which knows the underlying request distribution $\mathbf{p}$ and uses a fixed placement over the time horizon T, chosen to minimize the expected rate. This problem has been studied in the literature under the moniker 'coded caching under non-uniform requests' [4], [5], [12]. While characterizing the optimal scheme under a general request distribution is very challenging, there have been schemes proposed that are order-optimal, i.e., their expected server transmission rate is within a constant multiplicative[1] factor of the information-theoretic optimal. In particular, we will consider as benchmark[2] the scheme proposed in [4].

We illustrate the main idea of coded caching [2] via a simple toy example. Say we have $N = 2$ files A, B, and $K = 2$ users with caches C_1 and C_2 of size $M = 1$ each. Assume that the files are equally popular and thus the request distribution is $[0.5, 0.5]$ for both users. This makes all the possible request patterns AA, AB, BA, and BB equally likely. Let A_1, A_2 denote two halves of file A. As shown in Table I, when $[A_1, A_2]$ is stored in both the caches, the server broadcast transmission will be B for all requests except AA when no transmission is required ($\times$), bringing the average transmission size to 0.75. Similarly storing $[A_1, A_2]$ in C_1 and $[B_1, B_2]$ in C_2 will require server transmissions to be as per Table I. Note that when BA is requested, the server transmits the coded message $B \oplus A$. User 1 can retrieve B by performing an XOR operation on the message with the file A from its cache. Similarly, user 2 can retrieve A. The average transmission size for this storage profile also turns out to be 0.75. Finally, we consider the case where $[A_1, B_1]$ is stored in C_1 and $[A_2, B_2]$ is stored in C_2. Here, each of the transmissions is of size 0.5, giving a smaller average rate than above, while using coded server transmissions to serve all user requests.

The above illustrated ideas were generalized in [2] to any collection of parameters N, K, M. Furthermore, a decentralized coded caching scheme was proposed in [3] which enabled each user to randomly and independently sample M/N bits from each file and then use a coded delivery scheme as before. However, both [2], [3] assume all files to be equally popular, which does not always hold in practice. For non-uniform popularity distributions, we will consider as benchmark the order-optimal scheme proposed in [4], which uses the same placement and delivery policy as [3] but only on files that have popularity higher than a particular system-dependent threshold. More details are provided below.

The placement and delivery phases for the scheme in [4] are defined as follows:

i. *Placement*: All files whose underlying request probability is less than the threshold $1/(MK)$ are not stored in any cache. For the remaining, say N_1, most popular files, the disparity in their request probabilities is ignored and the placement policy proposed under the 'decentralized coded caching' scheme for uniformly popular files in [3] is used.
ii. *Delivery*: For unpopular files that have not been cached, the server serves the requests directly. For the N_1 popular files stored in the caches, the corresponding coded delivery policy from [3] is used.

For any underlying request distribution $\mathbf{p}$, the achievable normalized expected server transmission size, say K_o, in any slot for the scheme[3] above, denoted henceforth by $\pi^\star$, is shown in [4] to be at most

$$K_o \leq \left[\frac{N_1}{M} - 1\right]_+ + \min\left\{\sum_{i>N_1} Kp_i, \frac{N - N_1}{[M - N_1]_+} - 1\right\}. \tag{1}$$

Furthermore, the expected transmission rate for any caching scheme π is lower bounded [4] by

$$K_\pi \geq \max\left\{\frac{1}{29}\left[\frac{N_1}{M} - 1\right]_+, \frac{1}{58}\left[\sum_{i>N_1} Kp_i - 2\right]_+\right\} \triangleq K_{lb} \tag{2}$$

The upper and lower bounds are shown in [4] to be within a constant multiplicative and additive gap, independent of the system size.

[1] some results have an additional constant additive gap to the optimal also.

[2] Using an approximation algorithm as a benchmark for defining regret is common in the online learning literature when finding the optimal policy is intractable, see for example [24], [25].

[3] A more detailed description of the placement and delivery scheme from [4] is provided in Appendix VII-A

C_1	C_2	M_{AA}	M_{AB}	M_{BA}	M_{BB}	avg rate
$[A_1, A_2]$	$[A_1, A_2]$	$\times$	B	B	B	0.75
$[A_1, A_2]$	$[B_1, B_2]$	A	$\times$	$B \oplus A$	B	0.75
$[A_1, B_1]$	$[A_2, B_2]$	$A_2 \oplus A_1$	$A_2 \oplus B_1$	$B_2 \oplus A_1$	$B_2 \oplus B_1$	0.5

TABLE I
SUMMARY OF TRANSMISSION RATES FOR VARIOUS REQUEST PATTERNS AND CACHE CONTENTS

For any online policy π, we will define the *regret* over a time horizon T as the difference between its expected transmission size and that of the oracle scheme described above, i.e.,

$$R_\pi(T) = \mathcal{K}^\pi(T) - T.K_o = \sum_{t=1}^{T} \mathbb{E}[K_\pi(t) - K_o]. \quad (3)$$

This work aims to design an online policy with provably small regret w.r.t the oracle, without knowing the underlying distribution. In fact, as we demonstrate in Section III, a scheme which essentially has the same placement and delivery policy as $\pi^\star$ above, while using the empirical request distribution in each slot instead of the true underlying one for performing cache placement, has constant regret which does not scale with the time horizon T. We also provide a lower bound on the regret of any online policy in a restricted class.

Note that the placement scheme for an online policy can change over time slots and updating the cache contents will also incur some cost. For our proposed policy, we also provide an upper bound on the *switching cost* in terms of the overall number of time slots over any horizon T, say $S(T)$, where cache contents are updated.

For any integer $n \geq 1$, we will denote the set $\{1, 2, \ldots, n\}$ by $[n]$. Also, for integers $n_2 > n_1 \geq 1$, let $[n_1 : n_2]$ denote the subset $\{n_1, n_1 + 1, \ldots, n_2\}$.

III. ONLINE SCHEME AND ACHIEVABLE REGRET

In this section, we propose an online placement and delivery policy $\mathcal{P}$ which does not apriori know the underlying request distribution $\mathbf{p}$.

- Calculate the estimated popularity distribution $\{\hat{\mathbf{p}}^t\}_{t\geq 1}$ from requests seen until $t-1$.
- Perform placement and delivery for the estimated "popular" files as per [3].

Our policy tracks the empirical request probability distribution $\{\hat{\mathbf{p}}^t\}_{t\geq 1}$ based on the actual requests observed, and then assuming this to be the actual distribution, in each round performs caching according to the placement and delivery policy for the oracle scheme $\pi^\star$ as described before. That is, we use the placement and delivery policy in [3] to cache all files with empirical popularity at least $1/(KM)$ ("popular" files), while not storing the rest of the files in any cache and serving any requests for these files directly via the server. Our first result characterizes the regret of this online policy $\mathcal{P}$ with respect to the oracle policy $\pi^\star$, as defined in (3).

Theorem 1. *The regret of our proposed online caching policy $\mathcal{P}$ can be upper bounded as*

$$R_{\mathcal{P}}(T) \leq \min\{A, B\} + \underbrace{\frac{N}{M} - 1 - K_o}_{\textit{first step regret}}$$

where K_o is the expected (per slot) server transmission size for the oracle policy $\pi^\star$, $\Delta_i = \left|p_i - \frac{1}{KM}\right|$, $\Delta = \min_i \Delta_i$, and

$$A = \frac{(2 + \sum_{i=1}^{N} \Delta_i)(K - K_o)}{K\Delta^2}, B = \frac{4(K - K_o)}{K\Delta^2}.$$

The above result demonstrates that the proposed scheme $\mathcal{P}$ achieves a constant regret with respect to the oracle, which does not scale with the time horizon T.

Proof. Let $[N_1] = \{1, 2, \ldots, N_1\}$ denote the set of file indices for which the underlying request probability is greater than the threshold $1/(KM)$ (popular set). Then, we have

$$\Delta = \min_i \Delta_i = \min\left\{p_{N_1} - \frac{1}{KM}, \frac{1}{KM} - p_{N_1+1}\right\}$$

For each file W_i, define the empirical request probability at the start of time slot t as

$$\hat{p}_i^t = \begin{cases} \dfrac{\sum_{j=1}^{t-1}\sum_{k=1}^{K} \mathbb{1}\{r_k^j = i\}}{(t-1)K}, & t \geq 2, \\ \dfrac{1}{N}, & t = 1. \end{cases}$$

Note that in each slot, the oracle policy $\pi^\star$ caches the subset of files with indices in $[N_1]$ according to the decentralized coded caching in [3], whereas our policy applies the same scheme to the subset $\{i \in [N] : \hat{p}_i^t \geq \frac{1}{KM}\}$ at time $t > 1$. For $t = 1$, all the N files are included in the subset chosen to cache.

1) Upper bounding using Chernoff bounds: For each $t \geq 1$, let $\mathcal{G}^t$ denote the event that $\hat{p}_i^t > p_i - \Delta_i \ \forall\ i \in [N_1]$ and $\hat{p}_i^t < p_i + \Delta_i \ \forall\ i \in [N_1 + 1 : N]$. Under this event, we have that both the oracle and the proposed policy will cache the same files in slot t since $\forall\ j \in [N_1]$ we have $\hat{p}_j^t > p_j - \Delta_j \geq 1/(KM)$ and $\forall\ k \in [N_1 + 1 : N]$ we get $\hat{p}_k^t < p_k + \Delta_k \leq 1/(KM)$.

Lemma 1. *Let $\bar{\mathcal{G}}^t$ denote the complement of $\mathcal{G}^t$*

$$\mathbb{P}(\bar{\mathcal{G}}^t) \leq \sum_{i=1}^{N} \exp\left(-\frac{\delta_i^2 (t-1) K p_i}{2 + \delta_i}\right) \quad (4)$$

where $\delta_i = \Delta_i / p_i$ and total regret is bounded as

$$R_{\mathcal{P}}(T) \leq \frac{(2 + \sum_{i=1}^{N} \Delta_i)(K - K_o)}{K\Delta^2} + \frac{N}{M} - 1 - K_o. \quad (5)$$

The proof of this lemma can be found in Appendix VII-B of [23]

2) Upper bounding using the DKW inequality: We now present another way to upper bound the regret of the proposed scheme $\mathcal{P}$. Consider the event $\mathcal{H}^t : \max_i |\hat{p}_i^t - p_i| \leq \Delta$. It follows from [22, Lemma 2], which is an application of the Dvoretzky–Kiefer–Wolfowitz (DKW) inequality [26], that

$$\mathbb{P}(\bar{\mathcal{H}^t}) \leq 2e^{-(t-1)K\Delta^2/2}. \tag{6}$$

Under the event $\mathcal{H}^t$, we have that both the oracle policy and the proposed policy will take the same action since $\forall\ j \in [N_1]$ we have $\hat{p}_j^t \geq p_j - \Delta \geq p_{N_1} - \Delta \geq \frac{1}{KM}$ and $\forall\ k \in [N_1+1 : N]$ we get $\hat{p}_k^t \leq p_k + \Delta \leq p_{N_1+1} + \Delta \leq \frac{1}{KM}$. As before, we can bound the regret after the first time slot can be bounded as

$$\sum_{t=2}^{T}(K-K_o)\cdot\mathbb{P}(\bar{\mathcal{H}^t}) \overset{(a)}{\leq} 2(K-K_o)\sum_{t=2}^{T} e^{-(t-1)K\Delta^2/2}$$
$$\leq 2(K-K_o)\sum_{t=2}^{\infty} e^{-(t-1)K\Delta^2/2} \leq \frac{4(K-K_o)}{K\Delta^2}$$

where (a) follows from (6). Thus the overall regret for $\mathcal{P}$ can be bounded as

$$R_{\mathcal{P}}(T) \leq \frac{4(K-K_o)}{K\Delta^2} + \frac{N}{M} - 1 - K_o. \tag{7}$$

Combining (5) and (7) completes the proof. In order to express the bound in terms of only the problem parameters, one may lower bound K_o by K_{lb} □

A. Switching cost

Note that the placement scheme for an online policy can change over time slots, and updating the cache contents will also incur some cost. For our proposed policy, we now provide an upper bound on the expected switching cost in terms of the overall number of time slots over any horizon T, say $S(T)$, where cache contents are updated. In fact, we show that the expected switching cost is bounded above by an instance-dependent constant and does not scale with the horizon T.

Theorem 2. *For any horizon T, the expected switching cost for our proposed policy $\mathcal{P}$ can be upper-bounded as*

$$S(T) \leq 1 + \sum_{i\in[N_1]} \frac{1+\beta_i}{2K\Delta_i^2} + \sum_{i\in[N]\setminus[N_1]} \frac{\alpha_i+1}{2K\Delta_i^2}$$

where $[N_1] = \{1, 2, \ldots, N_1\}$ denotes the set of file indices for which the underlying request probability is greater than the threshold $1/(KM)$, and

$$\alpha_i = e^{2\Delta_i^2} \sum_{j=0}^{\lfloor \frac{1}{M} \rfloor} \binom{K}{j} p_i^j (1-p_i)^{K-j},$$
$$\beta_i = e^{2\Delta_i^2} \sum_{j=\lfloor \frac{1}{M} \rfloor+1}^{K} \binom{K}{j} p_i^j (1-p_i)^{K-j}.$$

Note that a cache update happens at time t when the estimated "popular set" changes. This happens either when an existing file is removed from the popular set (event $\mathcal{A}^t$), or a new file gets included in the popular set (event $\mathcal{B}^t$). Note that these events are not disjoint.

Lemma 2. *The probability of events $\mathcal{A}^t$ and $\mathcal{B}^t$ can be bounded as*

$$\mathbb{P}(\mathcal{A}^t) \leq \sum_{i\in[N_1]} e^{-2K(t-1)\Delta_i^2} + \sum_{i\in[N]\setminus[N_1]} \alpha_i e^{-2K(t-1)\Delta_i^2}$$

$$\mathbb{P}(\mathcal{B}^t) \leq \sum_{i\in[N_1]} \beta_i e^{-2K(t-1)\Delta_i^2} + \sum_{i\in[N]\setminus[N_1]} e^{-2K(t-1)\Delta_i^2}.$$

The proof of the above lemma can be found in Appendix VII-C in [23]. The proof of Theorem 2 can be found in Appendix VII-D in [23].

IV. Lower bound

Our next result is a lower bound on the expected regret of any online policy that does not apriori know the underlying request distribution. We restrict ourselves to a class, say $\mathcal{C}$, of policies that, in each round t, pick some subset of file indices S^t, and use the 'decentralized coded caching' placement and delivery policy in [3] to uniformly store files corresponding to this subset and serve their requests. If $|S^t| \geq M$, then the files outside this subset are not stored in any cache, and requests for such files are directly served by the server in the delivery phase. If $|S^t| < M$, the cache memory remaining after completely storing all files in S^t is used to perform uniform decentralized coded caching for the remaining files. Note that our proposed policy $\mathcal{P}$ from the previous section also belongs to this class $\mathcal{C}$.

For any policy $\pi \in \mathcal{C}$ and a given system instance $\mathcal{I} = (N, K, M, \mathbf{p})$, we consider the expected rate[4] incurred in round t by π to be

$$K_{S_\pi^t}(\mathcal{I}) = \begin{cases} \frac{|S_\pi^t|}{M} - 1 + \sum_{i\notin S_\pi^t} Kp_i & |S_\pi^t| > M \\ \frac{N-|S_\pi^t|}{M-|S_\pi^t|} - 1 & |S_\pi^t| \leq M \end{cases} \tag{8}$$

We will assume that N is even for brevity in our proof.

Theorem 3. *Consider any caching system with parameters N, M, K such that $N/K < M < N/2$. Fix any $0 < a < b$ with $\frac{1}{a} + \frac{1}{b} = \frac{1}{N}$ and $\frac{2}{b} < \frac{1}{KM} < \frac{2}{a}$. Then for any policy π in the class $\mathcal{C}$ described above, there exists a horizon T and an underlying request distribution (parameterized by a, b) such that*

$$R_\pi(T) \geq \frac{K(\frac{1}{KM} - \frac{2}{b})}{8\left(\frac{1}{a} - \frac{1}{b}\right)\log\left(\frac{b}{a}\right)}.$$

Proof. We choose $a < b$ such that $\frac{1}{a} + \frac{1}{b} = \frac{1}{N}$ and $\frac{1}{KM} \in (\frac{2}{b}, \frac{2}{a})$; such a pair will always exist for $KM > N$.

Next, we define

$$\mathbf{p} := \left\{ p_1 = \frac{2}{a} \cdots p_{N/2} = \frac{2}{a}, p_{N/2+1} = \frac{2}{b} \cdots p_N = \frac{2}{b} \right\}$$

[4]The rate expression used here is an approximation (in fact, an upper bound) for the actual expected rate of the scheme π, as follows from [3], [4]. The true expression is cumbersome, and hence we use this approximation to facilitate our analysis. However, a similar line of proof will apply there as well.

$$\mathbf{q} := \left\{ q_1 = \frac{2}{b} \cdots q_{N/2} = \frac{2}{b}, q_{N/2+1} = \frac{2}{a} \cdots q_N = \frac{2}{a} \right\}$$

and create two caching system instances $\mathcal{I}_p$, $\mathcal{I}_q$ with parameters N, K, M and underlying request distributions $\mathbf{p}$ and $\mathbf{q}$ respectively.

For an instance $\mathcal{I}$, define $S_1 = \left[\frac{N}{2}\right]$ and $S_2 = \left[\frac{N}{2} + 1 : N\right]$. A subset $S \subseteq [N]$ is said to be S_1-dominant if $\frac{|S \cap S_1|}{|S|} \geq \frac{1}{2}$, i.e., the majority of elements in S belong to S_1. Otherwise, it is said to be S_2-dominant (equivalently, $\frac{|S \cap S_2|}{|S|} > \frac{1}{2}$). We say that a caching subset S is "bad" if it is S_2-dominant when the underlying distribution is $\mathbf{p}$ or if it is S_1-dominant when the underlying distribution is $\mathbf{q}$.

Note that the oracle which knows the underlying request distribution will select files with a request probability larger than $1/KM$ for caching. That is, it will select S_1 for caching if the instance was $\mathcal{I}_p$ and S_2 if the instance was $\mathcal{I}_q$. Hence, the expected rate for the oracle in each slot is given by $K_o = \frac{N}{2M} - 1 + \frac{KN}{b}$ for both the instances $\mathcal{I}_p, \mathcal{I}_q$ from (8).

Now lemma 3 provides a lower bound on the regret incurred by any policy in a timeslot when it selects a "bad" subset of files to cache.

Lemma 3. *For any policy $\pi \in \mathcal{C}$ and either instance $\mathcal{I}_p$ or $\mathcal{I}_q$, let S be the set of files selected by the policy for caching for instance at some timeslot t. Let K_S denote the expected rate incurred in the slot. If S is "bad", then*

$$K_S - K_o \geq \Gamma, \text{ where } \Gamma \triangleq \frac{NK}{2}\left(\frac{1}{KM} - \frac{2}{b}\right)$$

This lemma is proved in Appendix VII-E in [23]

Now for any policy $\pi \in \mathcal{C}$ and some time horizon T, consider event $\mathcal{A}$ that the subsets of files selected for caching by π at different timeslots, denoted by $S_\pi^1, S_\pi^2, \cdots, S_\pi^T$, are S_2-dominant for at least half the slots. Formally,

$$\mathcal{A} = \left\{ \sum_{t=1}^{T} \mathbb{1}\left\{ \frac{|S_\pi^t \cap S_2|}{|S_\pi^t|} > \frac{1}{2} \right\} \geq \frac{T}{2} \right\}$$

Thus we have

$$\overline{\mathcal{A}} = \left\{ \sum_{t=1}^{T} \mathbb{1}\left\{ \frac{|S_\pi^t \cap S_1|}{|S_\pi^t|} \geq \frac{1}{2} \right\} \geq \frac{T}{2} \right\}$$

For instance $\mathcal{I}_p$, occurrence of event $\mathcal{A}$ for a policy π implies that S_π^t is "bad" for at least $T/2$ rounds. Similarly, for instance $\mathcal{I}_q$, occurrence of event $\overline{\mathcal{A}}$ for a policy π implies that S_π^t is "bad" for at least $T/2$ rounds.

Let $R_\pi^p(T)$ and $R_\pi^q(T)$ denote the regret incurred by π on the instances $\mathcal{I}_p$ and $\mathcal{I}_q$ with underlying request distributions $\mathbf{p}$ and $\mathbf{q}$ respectively. Under the event $\mathcal{A}$ for instance $\mathcal{I}_p$, the cache is "bad" for at least $\frac{T}{2}$ steps. From Lemma 3, the difference in the rate incurred by the policy π and the oracle is at least Γ in each of these $\frac{T}{2}$ steps. A similar argument holds for the instance $\mathcal{I}_p$ under the event $\overline{\mathcal{A}}$. Thus, we have

$$R_\pi^p(T) \geq \frac{T\Gamma}{2}\mathbb{P}_{\mathbf{p}}^\pi(\mathcal{A}), \quad R_\pi^q(T) \geq \frac{T\Gamma}{2}\mathbb{P}_{\mathbf{q}}^\pi(\overline{\mathcal{A}}). \tag{9}$$

Thus we obtain:

$$\begin{aligned} &\max\{R_\pi^p(T), R_\pi^q(T)\} \\ &\geq \frac{1}{2}(R_\pi^p(T) + R_\pi^q(T)) \geq \frac{T\Gamma}{2}(\mathbb{P}_{\mathbf{p}}^\pi(\mathcal{A})) + \mathbb{P}_{\mathbf{q}}^\pi(\overline{\mathcal{A}})) \\ &\overset{(a)}{\geq} \frac{T\Gamma}{4}\exp(-D_{KL}(\mathbb{P}_p^\pi || \mathbb{P}_q^\pi)) \\ &\overset{(b)}{=} \frac{T\Gamma}{4}\exp\left(-TN\left(\frac{1}{a} - \frac{1}{b}\right)\log\left(\frac{b}{a}\right)\right) \end{aligned}$$

where (a) follows from the Bretagnolle-Huber inequality [27, Theorem 14.2]; and (b) follows from Lemma 4 in Appendix VII-F in [23] which is based on a divergence decomposition result. The theorem statement then follows by choosing the horizon T which maximizes the expression on the right hand side of the above inequality.

V. Numerical experiments

In this section, we compare the performance of our policy using two benchmarks.

1) **Uniform coded caching**: This policy uses the same placement and delivery scheme used by us described in [3] for the cached files . However, it does not perform thresholding of any sort and caches all the files uniformly (N/M fraction of each file).
2) **Uncoded LFU**: This policy estimates the popularities after each time step and caches the top M files with the highest popularity estimates. Cached files are directly served, incurring 0 rate, and uncached files are entirely transmitted by the server.

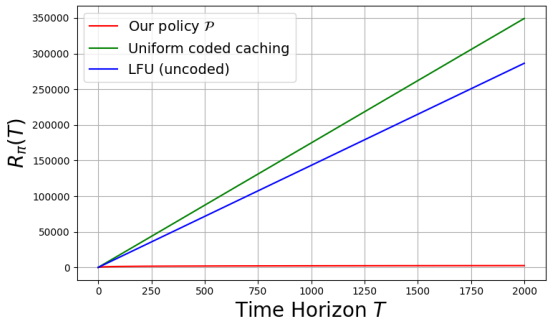
All the simulations are performed using estimated file popularities from the Movielens 1M dataset [28], which contains $\sim$ 1 million ratings from 6040 users on 3706 movies. Popularities are assigned using the fraction of ratings for a movie out of all ratings. The file requests were generated by sampling from this distribution and the regret for each policy was calculated as the difference of its rate from the oracle rate[5].

As seen in Figs. 2a-2c, our proposed policy, $\mathcal{P}$, outperforms the benchmarks. The uncoded LFU method performs significantly worse for large M, which emphasizes the superiority of coded schemes over uncoded ones. Moreover, the expected one-step regret is constant for both, the LFU and Uniform coding schemes. Hence, their regrets are seen to grow linearly with T. On the other hand, the regret for our policy $\mathcal{P}$ approaches a constant. This provides empirical verification for Theorem 1.

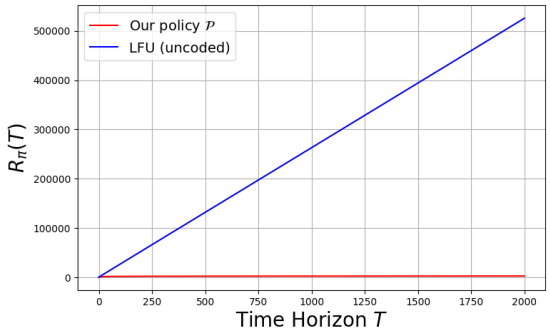
VI. Conclusions and Future Work

This work is a first step towards analyzing the widely studied coded caching framework through the lens of online learning. We consider the case where the underlying request distribution is apriori unknown and propose an online policy as well as study its regret with respect to an oracle that knows

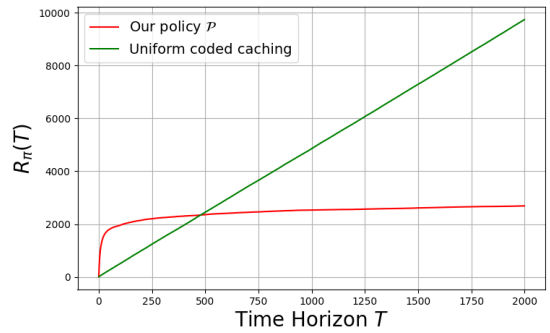
[5]We use the upper bound on the achievable oracle rate (described in section II) as this is known to be a good approximate for the actual rate for large K (K=400 was used) as shown in [4].



(a) K = 400, M = 10



(b) K = 400, M = 100



(c) K = 400, M = 100

Fig. 2. A comparison of benchmark policies against the proposed policy $\mathcal{P}$. In (a), our policy significantly outperforms Uniform Coded Caching [3] and uncoded LFU for small M. In (b), our policy is shown to outperform uncoded LFU for large M. Finally, in (c), we see that for large M, the regret for Uniform Coded Caching grows linearly with the time horizon T, whereas the regret of our policy $\mathcal{P}$ approaches a constant. This empirically confirms the bounds described in Theorem 1.

the underlying distribution and employs a well-known order-optimal placement and coded delivery strategy. We also bound the switching cost of this strategy and also discuss a lower bound on the regret of any online scheme in a restricted class.

There are several avenues for future work. Firstly, there are more complicated coded caching schemes other than the threshold-based policy considered here which are known to have better performance and can serve as the oracle benchmark for regret analysis. Secondly, while we restrict attention here to the case of stochastic requests from a stationary request distribution, there are several other interesting scenarios including a temporally varying request distribution and adversarial requests.

REFERENCES

[1] D. Wessels, Web caching. " O'Reilly Media, Inc.", 2001.

[2] M. A. Maddah-Ali and U. Niesen, "Fundamental limits of caching," IEEE Transactions on information theory, vol. 60, no. 5, 2014.

[3] M.-A. Maddah-Ali and U. Niesen, "Decentralized coded caching attains order-optimal memory-rate tradeoff," IEEE/ACM Transactions On Networking, vol. 23, no. 4, pp. 1029–1040, 2014.

[4] J. Zhang, X. Lin, and X. Wang, "Coded caching under arbitrary popularity distributions," in 2015 Information Theory and Applications Workshop (ITA), 2015, pp. 98–107.

[5] J. Hachem, N. Karamchandani, and S. N. Diggavi, "Coded caching for multi-level popularity and access," IEEE Transactions on Information Theory, vol. 63, no. 5, pp. 3108–3141, 2017.

[6] J. Hachem, N. Karamchandani, and S. Diggavi, "Multi-level coded caching," in 2014 IEEE international symposium on information theory. IEEE, 2014, pp. 56–60.

[7] J. Hachem, N. Karamchandani, and S. N. Diggavi, "Effect of number of users in multi-level coded caching," in 2015 IEEE international symposium on information theory (ISIT). IEEE, 2015.

[8] M. Ji, A. M. Tulino, J. Llorca, and G. Caire, "Order-optimal rate of caching and coded multicasting with random demands," IEEE Transactions on Information Theory, vol. 63, no. 6, 2017.

[9] N. Karamchandani, U. Niesen, M. A. Maddah-Ali, and S. N. Diggavi, "Hierarchical coded caching," IEEE Transactions on Information Theory, vol. 62, no. 6, pp. 3212–3229, 2016.

[10] C. Gurjarpadhye, J. Ravi, S. Kamath, B. K. Dey, and N. Karamchandani, "Fundamental limits of demand-private coded caching," IEEE Transactions on Information Theory, vol. 68, no. 6, 2022.

[11] V. Ravindrakumar, P. Panda, N. Karamchandani, and V. M. Prabhakaran, "Private coded caching," IEEE Transactions on Information Forensics and Security, vol. 13, no. 3, pp. 685–694, 2017.

[12] U. Niesen and M. A. Maddah-Ali, "Coded caching with nonuniform demands," IEEE Transactions on Information Theory, 2016.

[13] A. Cohen and T. Hazan, "Following the perturbed leader for online structured learning," in International Conference on Machine Learning. PMLR, 2015, pp. 1034–1042.

[14] J. Abernethy, C. Lee, A. Sinha, and A. Tewari, "Online linear optimization via smoothing," in Conference on Learning Theory. PMLR, 2014.

[15] M. Zinkevich, "Online convex programming and generalized infinitesimal gradient ascent," in Proceedings of the 20th international conference on machine learning (icml-03), 2003, pp. 928–936.

[16] G. S. Paschos, A. Destounis, L. Vigneri, and G. Iosifidis, "Learning to cache with no regrets," in IEEE INFOCOM 2019-IEEE Conference on Computer Communications. IEEE, 2019, pp. 235–243.

[17] G. S. Paschos, A. Destounis, and G. Iosifidis, "Online convex optimization for caching networks," IEEE/ACM Transactions on Networking, vol. 28, no. 2, pp. 625–638, 2020.

[18] T. S. Salem, G. Neglia, and S. Ioannidis, "No-regret caching via online mirror descent," in ICC 2021-IEEE International Conference on Communications. IEEE, 2021, pp. 1–6.

[19] R. Bhattacharjee, S. Banerjee, and A. Sinha, "Fundamental limits on the regret of online network-caching," Proceedings of the ACM on Measurement and Analysis of Computing Systems, 2020.

[20] D. Paria and A. Sinha, "Leadcache: Regret-optimal caching in networks," Advances in Neural Information Processing Systems, 2021.

[21] F. Zarin Faizal, P. Singh, N. Karamchandani, and S. Moharir, "Regret-optimal online caching for adversarial and stochastic arrivals," arXiv e-prints, pp. arXiv–2211, 2022.

[22] 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.

[23] A. authors, "On the regret of online coded caching," 2023. [Online]. Available: https://github.com/sheelfshah/OnlineCodedCaching

[24] Y. Azar, A. Fiat, and F. Fusco, "An α-regret analysis of adversarial bilateral trade," arXiv preprint arXiv:2210.06846, 2022.

[25] Y. Li, T. Si Salem, G. Neglia, and S. Ioannidis, "Online caching networks with adversarial guarantees," Proceedings of the ACM on Measurement and Analysis of Computing Systems, vol. 5, no. 3, pp. 1–39, 2021.

[26] A. Dvoretzky, J. Kiefer, and J. Wolfowitz, "Asymptotic minimax character of the sample distribution function and of the classical multinomial estimator," The Annals of Mathematical Statistics, pp. 642–669, 1956.

[27] T. Lattimore and C. Szepesvari, "Bandit algorithms," 2017. [Online]. Available: https://tor-lattimore.com/downloads/book/book.pdf

[28] F. M. Harper and J. A. Konstan, "The movielens datasets: History and context," ACM Trans. Interact. Intell. Syst., vol. 5, no. 4, dec 2015.

VII. Appendix

A. Placement and Delivery Scheme

Here we describe the placement and delivery scheme from [4]. The scheme essentially involves the use of the placement and delivery scheme described in [3] for a subset of files. This subset includes all the files with popularity $\geq \frac{1}{KM}$. The rest of the files are "unpopular" and are fully delivered by the server whenever requested.

1) Placement: In the cache placement stage at any time step t, once the selection of files S to be cached is determined, each user independently selects $\min\left\{|F|, \frac{M|F|}{|S|}\right\}$ bits randomly from the chosen file to store in their cache. Note that the set S for this policy consists of files for which the observed request probability until time t exceeds $\frac{1}{KM}$. The collection of files cached by the oracle remains constant as it knows the true request probability distribution, enabling it to select files that exceed the threshold of $\frac{1}{KM}$. If the number of files crossing this threshold is less than M, we store all the bits from the first M files and divide the remaining storage amongst the rest of the files uniformly.

2) Delivery: Let U_1 denote the set of users that request files from the cached set S, $|U_1| = K_4$

- For every subset $s \in U_1$ with $|s| \neq 0$ transmit $\bigoplus_{k\in s} V_{k,s\setminus\{k\}}$.
- For every request from U_1^c directly transmit the whole file ($|F|$ bits).

Here $V_{k,s\setminus\{k\}}$ denotes all the bits that are requested by user k, are present in the cache of all users in s and that are not stored in the caches of any other user in $U_1\setminus s$. $\bigoplus_{k\in s} V_{k,s\setminus\{k\}}$. denotes a XOR operation across all the users in s, i.e, a XOR across all the sets $V_{k,s\setminus\{k\}}$. Note that every user in s will be able to retrieve the bits designated to it from $\bigoplus_{k\in s} V_{k,s\setminus\{k\}}$ by repeatedly performing XOR operations using the bits present in its own cache. By construction, every bit requested by any user in U_1 has to be present in one of the sets $V_{k,s\setminus\{k\}}$.

B. Upper Bounds : Proof of Lemma 1

Let $\bar{\mathcal{E}}$ denote the complement of event $\mathcal{E}$. Then, we have

$$\begin{aligned}
\mathbb{P}(\bar{\mathcal{G}}^t) &= \mathbb{P}\left(\bigcup_{i=1}^{N_1}\{p_i - \hat{p}_i^t \geq \Delta_i\} \cup \bigcup_{i=N_1+1}^{N}\{\hat{p}_i^t - p_i \geq \Delta_i\}\right)\\
&\leq \sum_{i=1}^{N_1}\mathbb{P}(p_i - \hat{p}_i^t \geq \Delta_i) + \sum_{i=N_1+1}^{N}\mathbb{P}(\hat{p}_i^t - p_i \geq \Delta_i)\\
&= \sum_{i=1}^{N_1}\mathbb{P}\left(\hat{p}_i^t \leq p_i(1-\frac{\Delta_i}{p_i})\right) + \sum_{i=N_1+1}^{N}\mathbb{P}\left(\hat{p}_i^t \geq p_i(1+\frac{\Delta_i}{p_i})\right)\\
&\overset{(a)}{\leq} \sum_{i=1}^{N_1} e^{-\frac{\delta_i^2(t-1)Kp_i}{2}} + \sum_{i=N_1+1}^{N} e^{-\frac{\delta_i^2(t-1)Kp_i}{2+\delta_i}}\\
&\leq \sum_{i=1}^{N}\exp\left(-\frac{\delta_i^2(t-1)Kp_i}{2+\delta_i}\right) \qquad (10)
\end{aligned}$$

where $\delta_i = \Delta_i/p_i$ and (a) follows from the Chernoff bound. Next, we will evaluate the regret of our proposed policy. Recall that for the first time slot, we essentially assume all files to be equally popular and use the decentralized coded caching scheme from [3] over the set of all files. From [3], the expected server transmission size for this scheme is at most N/M, and hence the regret in the first slot is at most $N/M - K_o$. In any subsequent slot t, there will be no regret if the event $\mathcal{G}^t$ occurs since the proposed policy would be taking the same actions as the oracle policy. If $\mathcal{G}^t$ does not occur, then the regret in the slot will be at most $K - K_o$. We have

$$\begin{aligned}
&\sum_{t=2}^{T}(K-K_o)\cdot\mathbb{P}(\bar{\mathcal{G}}^t)\\
&\overset{(a)}{\leq} (K-K_o)\sum_{t=2}^{T}\sum_{i=1}^{N}\exp\left(-\frac{\delta_i^2}{2+\delta_i}(t-1)Kp_i\right)\\
&\leq (K-K_o)\sum_{i=1}^{N}\sum_{t=2}^{\infty}\exp\left(-\frac{\delta_i^2}{2+\delta_i}(t-1)Kp_i\right)\\
&= (K-K_o)\sum_{i=1}^{N}\left(\frac{\exp\left(-\frac{\delta_i^2}{2+\delta_i}Kp_i\right)}{1-\exp\left(-\frac{\delta_i^2}{2+\delta_i}Kp_i\right)}\right)\\
&\overset{(b)}{\leq} (K-K_o)\sum_{i=1}^{N}\left(\frac{2+\delta_i}{\delta_i^2 Kp_i}\right)\\
&= (K-K_o)\sum_{i=1}^{N}\left(\frac{2p_i+\Delta_i}{\Delta_i^2 K}\right) \leq (K-K_o)\cdot\frac{2+\sum_{i=1}^{N}\Delta_i}{\Delta^2 K}
\end{aligned}$$

where (a) follows from (10); and (b) follows since for each $x > 0$, $e^{-x}/(1-e^{-x}) \leq 1/x$. Thus the overall regret of our policy $\mathcal{P}$ is upper bounded as

$$R_{\mathcal{P}}(T) \leq \frac{(2+\sum_{i=1}^{N}\Delta_i)(K-K_o)}{K\Delta^2} + \frac{N}{M} - 1 - K_o. \quad (11)$$

C. Proof of Lemma 2

Proof.

$$\begin{aligned}
\mathbb{P}(\mathcal{A}^t) &= \mathbb{P}\left(\cup_{i=1}^{N}\{i\in\mathcal{N}_1^{t-1}, i\notin\mathcal{N}_1^t\}\right)\\
&\leq \sum_{i\in N}\mathbb{P}\left(\{i\in\mathcal{N}_1^{t-1}, i\notin\mathcal{N}_1^t\}\right)\\
&= \sum_{i\in N}\mathbb{P}\left(\hat{p}_i(t)\leq\frac{1}{KM}, \hat{p}_i(t-1)>\frac{1}{KM}\right)\\
&= \sum_{i\in[N_1]}\mathbb{P}\left(\hat{p}_i(t)\leq\frac{1}{KM}, \hat{p}_i(t-1)>\frac{1}{KM}\right)\\
&\quad + \sum_{i\in[N]\setminus[N_1]}\mathbb{P}\left(\hat{p}_i(t)\leq\frac{1}{KM}, \hat{p}_i(t-1)>\frac{1}{KM}\right)
\end{aligned}$$

where recall that $[N_1]$ denotes the set of popular files corresponding to the true underlying distribution. For the popular files, we have

$$\begin{aligned}
&\sum_{i\in[N_1]}\mathbb{P}\left(\hat{p}_i(t)\leq\frac{1}{KM}, \hat{p}_i(t-1)>\frac{1}{KM}\right)\\
&\leq \sum_{i\in[N_1]}\mathbb{P}\left(\hat{p}_i(t)\leq\frac{1}{KM}\right) \leq \sum_{i\in[N_1]} e^{-2K(t-1)\Delta_i^2}
\end{aligned}$$

where the last step follows from Hoeffding's inequality. For the unpopular files, we have

$$\begin{aligned}
&\sum_{i\in[N]\setminus[N_1]} \mathbb{P}\left(\hat{p}_i(t) \le \frac{1}{KM}, \hat{p}_i(t-1) > \frac{1}{KM}\right) \\
&= \sum_{i\in[N]\setminus[N_1]} \mathbb{P}\left(\hat{p}_i(t) < \frac{1}{KM} \middle| \hat{p}_i(t-1) \ge \frac{1}{KM}\right) \times \\
&\qquad\qquad \mathbb{P}\left(\hat{p}_i(t-1) \ge \frac{1}{KM}\right) \\
&\le \sum_{i\in[N]\setminus[N_1]} \mathbb{P}\left(\hat{p}_i(t) < \frac{1}{KM} \middle| \hat{p}_i(t-1) \ge \frac{1}{KM}\right) e^{-2K(t-2)\Delta_i^2} \\
&\le \sum_{i\in[N]\setminus[N_1]} \mathbb{P}\left(\sum_{k=1}^{K} \mathbb{1}\{r_k^t = i\} < 1/M\right) e^{-2K(t-2)\Delta_i^2} \\
&= \sum_{i\in[N]\setminus[N_1]} \alpha_i e^{-2K(t-1)\Delta_i^2}.
\end{aligned}$$

Thus overall, the probability of event $\mathcal{A}^t$ can be bounded as

$$\mathbb{P}(\mathcal{A}^t) \le \sum_{i\in[N_1]} e^{-2K(t-1)\Delta_i^2} + \sum_{i\in[N]\setminus[N_1]} \alpha_i e^{-2K(t-1)\Delta_i^2}$$

Similarly, the probability of event $\mathcal{B}^t$ can be bounded as

$$\mathbb{P}(\mathcal{B}^t) \le \sum_{i\in[N_1]} \beta_i e^{-2K(t-1)\Delta_i^2} + \sum_{i\in[N]\setminus[N_1]} e^{-2K(t-1)\Delta_i^2}.$$

□

D. Proof of Theorem 2

Proof. Recall that in the first slot, our policy includes all files in the popular set and thereafter only those files for which the empirical request probability is larger than the threshold $1/(KM)$. So we incur at most cost 1 at $t = 2$ and consider $t > 2$ hereafter. Let the total number of file requests for the file i before time t be denoted by L_i^{t-1}. Thus the empirical popularity $\hat{p}_i^t$ for file i will be $\frac{L_i^{t-1}}{K(t-1)}$. Depending on the number of requests observed for the file in time slot t, the updated empirical popularity $\hat{p}_i^{t+1}$ can range from $\frac{L_i^{t-1}}{Kt}$ to $\frac{L_i^{t-1}+K}{Kt}$. It is easy to verify that we always have $|\hat{p}_i(t+1) - \hat{p}_i(t)| \le \frac{1}{t}$. We will denote the estimated popular set of files at time t by $\mathcal{N}_1^t$. Thus, we have the following upper bound on the expected switch cost of our proposed policy $\mathcal{P}$:

$$\begin{aligned}
&1 + \sum_{t=2}^{T} (\mathbb{P}(\mathcal{A}^t) + \mathbb{P}(\mathcal{B}^t)) \\
&\le 1 + \sum_{t=2}^{T} \left(\sum_{i\in[N_1]} e^{-2K(t-1)\Delta_i^2} + \sum_{i\in[N]\setminus[N_1]} \alpha_i e^{-2K(t-1)\Delta_i^2} \right) \\
&+ \sum_{t=2}^{T} \left(\sum_{i\in[N]} \beta_i e^{-2K(t-1)\Delta_i^2} + \sum_{i\in[N]\setminus[N_1]} e^{-2K(t-1)\Delta_i^2} \right) \\
&\le 1 + \left(\sum_{i\in[N_1]} \frac{e^{-2K\Delta_i^2}}{1 - e^{-2K\Delta_i^2}} + \sum_{i\in[N]\setminus[N_1]} \alpha_i \frac{e^{-2K\Delta_i^2}}{1 - e^{-2K\Delta_i^2}} \right) \\
&+ \left(\sum_{i\in[N_1]} \beta_i \frac{e^{-2K\Delta_i^2}}{1 - e^{-2K\Delta_i^2}} + \sum_{i\in[N]\setminus[N_1]} \frac{e^{-2K\Delta_i^2}}{1 - e^{-2K\Delta_i^2}} \right) \\
&\le 1 + \left(\sum_{i\in[N_1]} \frac{1}{2K\Delta_i^2} + \sum_{i\in[N]\setminus[N_1]} \alpha_i \frac{1}{2K\Delta_i^2} \right) \\
&+ \left(\sum_{i\in[N_1]} \beta_i \frac{1}{2K\Delta_i^2} + \sum_{i\in[N]\setminus[N_1]} \frac{1}{2K\Delta_i^2} \right) \\
&= 1 + \sum_{i\in[N_1]} \frac{1+\beta_i}{2K\Delta_i^2} + \sum_{i\in[N]\setminus[N_1]} \frac{\alpha_i + 1}{2K\Delta_i^2}.
\end{aligned}$$

□

E. Proof of Lemma 3

Proof. Consider instance $\mathcal{I}_p$ and the case where $|S| > M$. Then,

$$\begin{aligned}
K_S(\mathcal{I}_p) &= \frac{|S|}{M} + K \sum_{i\notin S} p_i - 1 \\
&\overset{(a)}{\ge} \frac{|S|}{M} + K\frac{2}{b}\left\lfloor \frac{N-|S|}{2} \right\rfloor + K\frac{2}{a}\left\lceil \frac{N-|S|}{2} \right\rceil - 1 \\
&\overset{(b)}{\ge} \frac{|S|}{M} + \frac{K(N-|S|)}{N} - 1 \\
&= \frac{|S|(N-KM) + KMN}{NM} - 1 \\
&\overset{(c)}{\ge} \frac{N}{M} - 1
\end{aligned}$$

And when $|S| \le M$

$$K_S(\mathcal{I}_p) = \frac{N-|S|}{M-|S|} - 1 \ge \frac{N}{M} - 1$$

Therefore, in both the cases we have

$$\begin{aligned}
K_S(\mathcal{I}_p) - K_o &\ge \frac{N}{M} - 1 - \left(\frac{N}{2M} - 1 + \frac{KN}{b}\right) \\
&= \frac{N}{2M} - \frac{KN}{b} = \frac{NK}{2}\left(\frac{1}{KM} - \frac{2}{b}\right)
\end{aligned}$$

where (a) follows from observing that $\frac{2}{a} > \frac{2}{b}$ and $\frac{|S^c \cap S_1|}{|S^c|} \ge \frac{1}{2}$ where $S^c = \{1, 2, \cdots, N\} \setminus S$. Since $|S^c| = N - |S|$, at least $\frac{N-|S|}{2}$ files of S^c have request probability $\frac{2}{a}$ and the rest have a request probability of at least $\frac{2}{b}$. (b) follows immediately if $N - |S|$ is even. If $N - |S|$ is odd, then $\frac{2}{b}\lfloor\frac{N-|S|}{2}\rfloor + \frac{2}{a}\lceil\frac{N-|S|}{2}\rceil = \frac{N-|S|-1}{2} \cdot \frac{2}{N} + \frac{2}{a} \ge \frac{N-|S|-1}{N} + \frac{1}{a} + \frac{1}{b} = \frac{N-|S|}{N}$. Finally, (c) follows by observing that $|S|(N-KM)$ is minimum when $|S| = N$ since $N - KM < 0$ and $0 \le |S| \le N$.

A similar argument holds for $\mathcal{I}_q$ as well. □

F. Divergence Calculation

Lemma 4. *The KL-divergence between the probability measures $\mathbb{P}_p^\pi$ and $\mathbb{P}_q^\pi$ induced by policy π on the set of all possible*

history sequences $s \in \mathcal{S}$ with a horizon T under instances $\mathcal{I}_p$ and $\mathcal{I}_q$, respectively, is

$$D_{KL}(\mathbb{P}_p^\pi || \mathbb{P}_q^\pi) = T D_{KL}(p||q)$$

where $D_{KL}(p||q)$ is the divergence between the underlying popularity distributions p and q which is given by

$$D_{KL}(p||q) = \left(\frac{N}{a} - \frac{N}{b}\right) \log\left(\frac{b}{a}\right)$$

Proof. Consider the interleaved sequence s of observed file requests (X_i) and caching decisions (C_i) $\{C_1, X_1, C_2 \cdots C_T, X_T\}$ and let $\mathcal{S}$ denote the set of all possible sequences. $\mathbb{P}_p^\pi$ and $\mathbb{P}_q^\pi$ denote the probability measures on these observed sequences under their respective instances given the policy π. Also, note that the observed file requests at any point in time are only dependent on the underlying popularity distribution and are independent of whatever is present in the cache.

$$\begin{aligned}\mathbb{P}_p^\pi(s) &= \prod_{i=1}^{T} \mathbb{P}_p(X_i|C_i, X_{i-1}, C_{i-1} \cdots C_1, \pi) \\ &\quad \times \mathbb{P}(C_i|X_{i-1}, C_{i-1} \cdots C_1, \pi) \\ &= \prod_{i=1}^{T} \mathbb{P}_p(X_i) \times \mathbb{P}(C_i|\mathcal{H}(i), \pi)\end{aligned}$$

Where $\mathcal{H}(t)$ denotes the history i.e., the set of all cache decisions and request patterns seen up to time $t-1$. Similarly one has

$$\mathbb{P}_q^\pi(s) = \prod_{i=1}^{T} \mathbb{P}_q(X_i) \times \mathbb{P}(C_i|\mathcal{H}(i), \pi)$$

The divergence $D_{KL}(\mathbb{P}_p^\pi || \mathbb{P}_q^\pi)$ can now be written as

$$\begin{aligned}&= \sum_{s\in\mathcal{S}} \mathbb{P}_p^\pi(s) \log\left(\frac{\mathbb{P}_p^\pi(s)}{\mathbb{P}_q^\pi(s)}\right) \\ &= \sum_{s\in\mathcal{S}} \mathbb{P}_p^\pi(s) \log\left(\frac{\prod_{i=1}^{T} \mathbb{P}_p(X_i) \times \mathbb{P}(C_i|\mathcal{H}(i), \pi)}{\prod_{i=1}^{T} \mathbb{P}_q(X_i) \times \mathbb{P}(C_i|\mathcal{H}(i), \pi)}\right) \\ &= \sum_{s\in\mathcal{S}} \mathbb{P}_p^\pi(s) \log\left(\frac{\prod_{i=1}^{T} \mathbb{P}_p(X_i)}{\prod_{i=1}^{T} \mathbb{P}_q(X_i)}\right) \\ &= \sum_{s\in\mathcal{S}} \mathbb{P}_p^\pi(s) \sum_{i=1}^{T} \log\left(\frac{\mathbb{P}_p(X_i)}{\mathbb{P}_q(X_i)}\right) \\ &= \sum_{i=1}^{T} \sum_{s\in\mathcal{S}} \mathbb{P}_p^\pi(X_i|s \setminus X_i) \mathbb{P}_p^\pi(s \setminus X_i) \log\left(\frac{\mathbb{P}_p(X_i)}{\mathbb{P}_q(X_i)}\right) \\ &= \sum_{i=1}^{T} \sum_{s\in\mathcal{S}_i} \mathbb{P}_p^\pi(s \setminus X_i) \sum_{X_i} \mathbb{P}_p^\pi(X_i|s \setminus X_i) \log\left(\frac{\mathbb{P}_p(X_i)}{\mathbb{P}_q(X_i)}\right) \\ &= \sum_{i=1}^{T} \sum_{s\in\mathcal{S}_i} \mathbb{P}_p^\pi(s \setminus X_i) \sum_{X_i} \mathbb{P}_p(X_i) \log\left(\frac{\mathbb{P}_p(X_i)}{\mathbb{P}_q(X_i)}\right) \\ &= \sum_{i=1}^{T} \sum_{s\in\mathcal{S}_i} \mathbb{P}_p^\pi(s \setminus X_i) D_{KL}(p||q) \\ &= \sum_{i=1}^{T} D_{KL}(p||q) = T D_{KL}(p||q)\end{aligned}$$

where $\mathcal{S}_i$ is the set of all possible sequences $\{C_1, X_1, C_2 \cdots C_i, C_{i+1}, X_{i+1}, \cdots, C_T, X_T\}$ and $\mathcal{X}_i$ is the set of all possible request vectors (i.e. $[N]^K$). (i.e. $\mathcal{S} = \mathcal{S}_i \times \mathcal{X}_i$)

$D_{KL}(p||q)$ can be written as

$$\begin{aligned}D_{KL}(p||q) &= \sum_{i=1}^{N} p_i \log\left(\frac{p_i}{q_i}\right) \\ &= \sum_{i=1}^{\frac{N}{2}} \frac{2}{a} \log\left(\frac{b}{a}\right) + \sum_{\frac{N}{2}+1}^{N} \frac{2}{b} \log\left(\frac{a}{b}\right) \\ &= \left(\frac{N}{a} - \frac{N}{b}\right) \log\left(\frac{b}{a}\right)\end{aligned}$$

□