Fundamental Limits of Online Network-Caching

Rajarshi Bhattacharjee  
Dept. of Electrical Engineering  
IIT Madras  
brajarshi91@gmail.com

Subhankar Banerjee  
Dept. of Electrical Engineering  
IIT Madras  
ee16s048@ee.iitm.ac.in

Abhishek Sinha  
Dept. of Electrical Engineering  
IIT Madras  
abhishek.sinha@ee.iitm.ac.in

ABSTRACT
Optimal caching of files in a content distribution network (CDN) is a problem of fundamental and growing commercial interest. Although many different caching algorithms are in use today, the fundamental performance limits of network caching algorithms from an online learning point-of-view remain poorly understood to date. In this paper, we resolve this question in the following two settings: (1) a single user connected to a single cache, and (2) a set of users and a set of caches interconnected through a bipartite network. Recently, an online gradient-based coded caching policy was shown to enjoy sub-linear regret. However, due to the lack of known regret lower bounds, the question of the optimality of the proposed policy was left open. In this paper, we settle this question by deriving tight non-asymptotic regret lower bounds in both of the above settings. In addition to that, we propose a new Follow-the-Perturbed-Leader-based uncoded caching policy with near-optimal regret. Technically, the lower-bounds are obtained by relating the online caching problem to the classic probabilistic paradigm of balls-into-bins. Our proofs make extensive use of a new result on the expected load in the most populated half of the bins, which might also be of independent interest. We evaluate the performance of the caching policies by experimenting with the popular MovieLens dataset and conclude the paper with design recommendations and a list of open problems.

CCS CONCEPTS
- Networks  →  Network performance analysis;  
- Mathematics of computing  →  Probabilistic algorithms.

KEYWORDS
network-caching, online algorithms, regret bounds, fundamental limits

ACM Reference Format:
Rajarshi Bhattacharjee, Subhankar Banerjee, and Abhishek Sinha. 2020. Fundamental Limits of Online Network-Caching. In Sigmetrics ’20, June 08–12, 2020, Boston, MA. ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION AND RELATED WORK
The classical caching problem, which seeks to make popular contents quickly accessible by prefetching them in low-latency storage, has been extensively studied in the literature. The core idea of caching has been used in many diverse domains, including improving the CPU paging performance via $L_1/L_2$ caches [1], web-caching by Content Distribution Networks [2–4], and low-latency wireless video delivery through Femtocaching [5]. With the exponential growth of internet video traffic and the advent of new services consuming high bandwidth, such as augmented and virtual reality (AR/VR), the importance of caching for ensuring the quality of service (QoS) is on the rise [6]. Top CDN providers, such as Amazon AWS and Microsoft Azure, now offer caching as a service [7, 8].

Several caching algorithms have been proposed in the literature. The MIN algorithm [9] is an optimal offline caching policy which assumes that the entire file request sequence is known non-causally in advance. MIN is often used as a benchmark for comparing the performance of the online caching policies. Among the online policies, the Least Recently Used policy (LRU), the Least Frequently Used policy (LFU) [10], the FIFO policy [11], and the online coded caching policy [12] have been studied extensively. However, the performance guarantees available for most of the online caching policies are highly contingent upon some a priori assumptions on the generative model of the file request sequence [13, 14]. The paper [15] analyzed the performance of the LRU caching policy with an i.i.d. file request sequence, also known as the Independent Reference Model [16]. Under a Markovian assumption, the paper [12] shows that a coded caching policy out-performs the LRU policy. The paper [17] develops a unified framework for analyzing a number of popular caching policies, again with a stationary file request model. On a different line of work, the papers [18–21] derive information-theoretic lower bounds and efficient caching, computing, and coding schemes to facilitate bandwidth-efficient delivery of the cached files to the users.

With frequent addition of new content to the library, mobility of the users, Femtocaching with small caches, and change in the popularity distribution with time, the assumption of stationary file popularity barely holds in practice [22]. This prompts us to consider the problem of caching from an online learning point-of-view with no a priori statistical assumption on the file request sequence. Our work is inspired by the recent paper [23], which describes an online gradient-based coded caching policy (OGA), and proves a sub-linear regret upper-bound for the same. Interestingly, they also show that popular uncoded caching policies, such as LRU, LFU, and FIFO, suffer from linear regrets in the worst case. In fact, no uncoded caching policy with a sub-linear regret is known previously in the literature. More seriously, no regret lower bound is known for the network-caching problem.
In contrast to the multi-armed bandits setting [24–28], relatively few results are known for the regret lower bounds for online convex optimization problems. Technically, the network-caching problem is an instance of an online convex optimization problem with a piecewise linear reward function and polytope constraints. The paper [29] establishes a minimax regret lower bound for linear cost functions with hyper ball constraints. A regret lower bound for the unconstrained linear cost functions has been obtained in [30]. The papers [31, 32] prove logarithmic regret bounds for online stochastic strongly convex problems. However, to the best of our knowledge, with the exception of [23], the regret for a linear cost function with a simplex and several box constraints, which arise in the context of the single cache problem, has not been studied before. Moreover, the problem of lower bounding the regret for a piecewise linear cost function with polytope constraints, which arise in the context network-caching, is completely open.

The above considerations inspire us to ask the following two questions in this paper:

**Question 1.** What is the fundamental performance limit of all online caching policies regardless of their operational constraints or computational complexity?

**Question 2.** Can a simple, distributed network-caching strategy be designed which meets the above fundamental limit?

In answering Question 1, we derive universal regret lower bounds that also apply to computationally intensive caching policies, which can completely change the profile of the cached contents at every time slot. Surprisingly enough, we answer Question 2 in the affirmative. In particular, our matching upper and lower bounds reveal that a simple gradient-based incremental coded caching policy is regret-optimal. Moreover, we propose a new Follow-the-Perturbed-Leader-based uncoded caching policy that has near-optimal regret. Hence, one of the key take-away points from this paper is that there exist computationally cheap caching policies that perform excellently in an online setting even with adversarial request sequence.

**Our contributions:** In the process of answering the above questions, we make the following key technical contributions:

1. **Lower bound for a single cache:** In Theorem (2), we prove a tight non-asymptotic regret lower bound for the single-cache problem. This result improves upon the previously known asymptotic regret lower bound in [23], which can be arbitrarily loose for a sufficiently large library size.

2. **Lower bounds for caching networks:** In Theorems (6) and (7), we derive non-asymptotic sub-linear regret lower bounds for bipartite caching networks. We also show that the lower bounds are tight within constant factors. To the best of our knowledge, this is the first known regret lower bound for piecewise linear functions with polytope constraints. Hence, our results also contribute to the growing literature on online convex optimization.

3. **Near-optimal uncoded caching policy:** Although the above lower bounds certify the optimality of the gradient-based coded caching policy of [23], it was an open problem whether one could achieve the optimal regret without coding. In Algorithm 1, we propose a simple uncoded caching policy based on the Follow-the-Perturbed-Leader (FTPL) paradigm. Theorem 3 shows that the FTPL policy has near-optimal regret.

4. **New proof techniques:** Technically, the regret lower bounds are established by relating the online caching problem to the classic probabilistic setup of balls-into-bins via Lemma 1. In this Lemma, we derive a non-asymptotic lower bound to the expected total load in the most populated bins when $m$ balls are randomly thrown into $2n$ bins. For our lower bound proofs, we are particularly interested in the regime $m \gg n$. It is to be noted that classical results on randomized load balancing, such as [33–35], do not apply to this setting because they apply only to the regime where $m = O(n)$. It is to be noted that the paper [36] derives an asymptotic high-probability bound for the maximum load for various asymptotic regimes of $m$ and $n$. However, since we are primarily interested in the non-asymptotic expected value of Max-Load, the results of [36] do not suffice for our purpose. Consequently, we tackle this problem from the first principles, culminating in Lemma 1. To the best of our knowledge, this is the first paper where a connection between online learning and the framework of balls-into-bins has been explicitly brought out and exploited in proving regret lower bounds.

5. **Numerical experiments.** In Section 4, we compare the performance of different caching policies using the popular MovieLens 1 M dataset [37]. Our experiments reveal that the proposed FTPL policy beats other competitive caching policies in terms of long-term average regret.

## 2 SINGLE CACHE

In this section, we begin our investigation with the single cache problem. We establish a key technical lemma on the balls-into-bins problem, which is used in all of our lower bound proofs. Our analysis in this section improves upon the best-known regret lower bound for the single caches [23] and paves the way for analyzing the bipartite caching problem in Section 3.

### 2.1 System Model

In the classical caching problem with a single cache, there is a library of $N$ distinct files. A cache with a limited storage capacity can store at most $C$ files at any time slot. In practice, the cache capacity is significantly smaller compared to the entire library size ($i.e., C \ll N$). As an example, we may think of caching movie files in the Netflix data center, where the library size $N$ increases every day as new movies are released, but the physical cache-size in the Netflix data center remains constant on the relevant time-scale. Time is slotted, and a user may request at most one file at a time. The file requests at time slot $t$ is represented by an $N$-dimensional binary vector $\mathbf{x}_t \in \{0, 1\}^N$, where $x_{tf} = 1$ if the $f$th file is requested by the user at time $t$, and is zero otherwise (one-hot encoding). Following an online caching policy $\pi$, files are cached at every time slot before the request for that slot arrives. We do not make any statistical assumption on the file request sequence $\{x_{tf}\}$.

**Coded Caching:** In this paper, we consider both coded and uncoded caching. In the classical uncoded caching, complete files are
We assume that the cumulative reward over a time horizon $k$ will see that coding offers distinct advantages for network caching. The function $q$ denoted by $\pi$ the request vector $q$, rewards obtained at every slot up to time $t$, are additive in nature, which may be obtained by summing up the average hit rate $i.e.,$

$$Q(\{x_t\}_T, \{y_t\}_T) = \sum_{t=1}^T q(x_t, y_t).$$  

For the case of a single cache, we define the one-slot reward to be the amount of the requests successfully served by the cache, i.e.,

$$q(x_t, y_t) \equiv x_t \cdot y_t.$$  

A generalized definition of the above one-slot reward function applicable to caching networks will be given in Section 3.

2.3 Regret

Since we do not make any assumption on the user-generated file request sequence $\{x_t\}_{t\geq1}$, it is futile to attempt to optimize the total reward (i.e., hit rate). This is because, at any slot, an omniscient adversary can always request a file which is not present in the cache, thus yielding a total of zero hits. To obtain a non-trivial performance measure, we cast the caching problem into the framework of online learning. This prompts us to compare the performance of any policy with the best offline stationary optimal policy [41]. Let the vector $y^*$ denote any fixed stationary cache configuration vector. The vector $y^*$ may be selected offline after seeing the entire request sequence $\{x_t\}_{t=1}^T$. Following the usual convention in the online learning literature, we define the regret $R_T^\pi(\{x_t\}_T)$ for any request sequence $\{x_t\}_T$ to be the difference in the reward obtained by the best stationary caching configuration $y^*$ and that of the online policy $\pi$. Mathematically$^2$,

$$R_T^\pi(\{x_t\}_T) := \sup_{y \in \mathcal{Y}} \left( Q(\{x_t\}_T, \{y^*\}_T) - Q(\{x_t\}_T, \{y_t\}_T) \right).$$  

The regret $R_T^\pi$ of any caching policy $\pi$ up to time $T$ is defined to be the maximum regret over all admissible request sequences, i.e.,

$$R_T^\pi := \sup_{\{x_t\}_T} R_T^\pi(\{x_t\}_T).$$  

2.4 Regret lower bounds - preliminaries

Using the simple observation that the maximum of a set of real numbers is at least equal to their average, for any joint probability distribution $P(\{X_t\}_{t=1}^T)$ on the file request sequence $\{X_t\}_T$, the regret in Eqn. (5) may be lower bounded as follows:

$$R_T^\pi \geq \mathbb{E}_{\{X_t\}_T} R_T^\pi(\{X_t\}_T).$$  

The joint distribution $P(\cdot)$ needs to be chosen carefully to ensure the tractability of evaluating the expectation in (6), as well as the tightness of the resulting bound. The above technique is known as the probabilistic method popularized by Erdős [42]. In all of our proofs, we lower bound the quantity in Eqn. (4) by a suitable binary cache configuration vector $y^*$ (thus $y^*$ corresponds to uncoded caching). As a consequence, all of our lower bounds remain valid in both uncoded and coded caching.

2.5 Regret lower bound for unit-sized cache

We first consider a single cache of unit capacity and derive a universal non-asymptotic regret bound. As we will see in the sequel, understanding this special case lays the foundation to our subsequent analysis of caching networks serving multiple users.

---

$^1$Whenever the caching policy $\pi$ is clear from the context, we will drop the superscript $\pi$ to simplify the notation.

$^2$Recall that the cache configuration sequence $\{y_t\}_{t\geq1}$ is determined by the policy $\pi$. 

---
The regret $R_T^π$ of any online caching policy, for a library-size $N = 2$ and cache-capacity $C = 1$, is lower bounded as:

$$R_T^π ≥ \sqrt{\frac{T}{2π} - \frac{1}{2\sqrt{2}/π T}}, \forall T ≥ 1.$$  

**Proof.** Denote the cache configuration selected by the caching policy $π$ at slot $t$ by the vector $y_t = (y_t, 1 - y_t)^T$, where $y_t$ denotes the fraction of File 1 cached by the policy $π$ ($0 ≤ y_t ≤ 1$). Recall the definition of regret $R_T^π$ in this context:

$$R_T^π = \sup_{\{x_t\}_1^T} \sup_{y \in Y} \left\{ y^* \cdot \sum_{t=1}^T x_t - \sum_{t=1}^T (y_t \cdot x_t) \right\},$$  

where the set of all admissible configurations $Y$ is given by Eqn. (1). Denote the file request vector at slot $t$ by $x_t = (w_t, 1 - w_t)^T$, where $w_t \in \{0, 1\}$. It can be easily seen that for a given file request sequence $\{x_t\}_1^T$, an optimal choice of the fixed cache configuration vector $y^*$ in Eqn. (7) is given as follows:

$$y^* = \begin{cases} (1, 0)^T, & \text{if } \sum_{t=1}^T w_t ≥ T/2, \\ (0, 1)^T, & \text{if } \sum_{t=1}^T w_t < T/2. \end{cases}$$

To prove a universal regret lower bound, we need to show the existence of an adversarial file request sequence $\{x_t\}_1^T$ under which the caching policy $π$ performs poorly. Towards this, let $\{W_t\}_1^T \geq 1$ be a sequence of i.i.d. uniform Bernoulli random variables such that $\mathbb{P}(W_t = 0) = \mathbb{P}(W_t = 1) = 1/2$. Construct a random file request sequence $X_t = (W_t, 1 - W_t)^T$. The regret incurred for the sequence $\{X_t\}_1^T$ may be obtained from Eqn. (7) as:

$$R_T^π(\{X_t\}_1^T) = \max \left\{ \sum_{t=1}^T W_t, T - \sum_{t=1}^T W_t \right\} - \sum_{t=1}^T (y_t W_t + (1 - y_t)(1 - W_t))$$

$$= \max \left\{ Z, T - Z \right\} - \sum_{t=1}^T (2y_t W_t + 1 - y_t - W_t),$$

where the r.v. $Z ≡ \sum_{t=1}^T W_t$, being the summation of $T$ i.i.d. uniform Bernoulli variables, is Binomially distributed with parameter $(T, 1/2)$. Using linearity of expectation, we can write

$$\mathbb{E}(\{X_t\}_1^T)^T R_T^π(\{X_t\}_1^T) = \mathbb{E}(\max \left\{ Z, T - Z \right\}) - T/2,$$

where we have used the fact that $\mathbb{E}(W_t) = 1/2$ and the caching decision $y_t$ is independent of the incoming request $X_t$. Observe that, we can write

$$\max \left\{ Z, T - Z \right\} = \frac{T}{2} + |Z - T/2|. $$

Thus, combining Eqns. (8) and (9), we have

$$\mathbb{E}(\{X_t\}_1^T)^T R_T^π(\{X_t\}_1^T) = \mathbb{E}[Z - T/2].$$

The mean absolute deviation for a symmetric binomial random variable may be computed in closed form by using De Moivre’s formula ([43], Eqn. (11)) as follows:

$$\mathbb{E}[Z - T/2] = \frac{1}{2} \left[ \left( \frac{T}{2} + 1 \right) \left( \frac{T}{2} - 1 \right) \right].$$

Eqn. (11), in combination with non-asymptotic form of Stirling’s formula ([44]), yields the following non-asymptotic lower bound

$$\mathbb{E}[Z - T/2] ≥ \frac{T}{2\pi} - \frac{1}{\sqrt{2\pi T}}, \forall T ≥ 1.$$  

(12)

For details of the above calculations, please refer to Appendix 7.1. Equation (12), coupled with Eqn. (10), shows the existence of a file request sequence $\{x_t\}_1^T$ such that

$$R_T^π(\{x_t\}_1^T) ≥ \frac{T}{2\pi} - \frac{1}{\sqrt{2\pi T}}, \forall T ≥ 1.$$  

□

**Remarks:** It can be seen that the above proof and the lower bound in Theorem 1 continue to hold even if we let the library size to be $N ≥ 2$. This observation follows by constructing a randomized file request sequence where the first two files are requested with probability $1/2$ each and other files are requested with zero probability.

### 2.6 Lower bound for caches of arbitrary size

We now extend the previous result to caches with arbitrary size $C ≥ 1$. This extension is non-trivial. We will see in the sequel that our analysis naturally leads us to investigate a random variable arising in connection with the classic probabilistic framework of {	extit{balls-into-bins}}, where a number of balls are thrown uniformly and independently at random to some bins [36]. The following lemma, which might be of independent interest, gives a non-asymptotic lower bound to the total number of balls in the most populated half of the bins.

**Lemma 1 (Total occupancy in the most popular half).**

Suppose that $T$ balls are thrown independently and uniformly at random into $2C$ bins. Let the random variable $M_{C}(T)$ denote the number of balls in the most populated $C$ bins. Then

$$\mathbb{E}(M_{C}(T)) ≥ \frac{T}{2} + \sqrt{\frac{CT}{2\pi}} - \left( \frac{\sqrt{2} + 1}{2} \right)^{3/2} - \sqrt{\frac{C^2}{2\pi T}}.$$  

**Proof outline:** The proof proceeds by pairing up the bins to form {	extit{super bins}} (see Fig. 4), and then selects the most-occupied bin in each super bin to obtain a lower bound on $M_{C}(T)$. Finally, we conclude the proof by appealing to the mean-deviation bound in Eqn. (12). Please refer to Section 6.1 for the complete proof of Lemma 1.

To improve readability, in the rest of the paper we will rephrase the above bound as

$$\mathbb{E}(M_{C}(T)) ≥ \frac{T}{2} + \sqrt{\frac{CT}{2\pi}} - \Theta\left( \frac{1}{\sqrt{T}} \right),$$

with the understanding that an explicit form of the lower order terms may be obtained by using Lemma 1, if required. The following Theorem is the main result of this section.
Theorem 2 (Lower bound for a single cache of arbitrary capacity). The regret $R_π^T$ of any online caching policy $π$, for a library size $N$ and cache capacity $C$ with $N \geq 2C$, is lower bounded as

$$R_π^T \geq \sqrt{\frac{CT}{2\pi} - \Theta\left(\frac{1}{\sqrt{T}}\right)} \quad \forall T \geq 1. \quad (13)$$

Proof outline: The proof proceeds along the lines of Theorem 1, where the $t^{th}$ file requested $X_t$ is chosen independently and uniformly at random from the first $2C$ files from the library. Then, we show that the reward accrued by the best fixed cache configuration $y^*$ corresponds (in distribution) to the total number of balls in the most populated half of the bins. We then conclude the proof by appealing to Lemma 1. Please refer to Section 6.2 for the complete proof of Theorem 2.

Comparison with Theorem 1 of [23]: In the single cache setting, the paper [23] establishes a rather loose asymptotic regret lower bound of $\sqrt{\frac{C}{N\pi} \sqrt{CT}}$, which could be arbitrarily smaller than the lower bound given in Eqn. (13) when the library size $N$ is sufficiently large. Theorem 2 improves the result in [23] in two ways. First, it proves a regret lower bound that is independent of $N$. As a consequence, we will soon see that it implies that the gradient-based coded policy of [23] is regret-optimal up to a constant factor. Theorem 2 also applies near optimality of the uncoded FTPL policy described in the next section. Second, unlike the regret lower bound in [23], the bound in Eqn. (2) is non-asymptotic, thus giving a valid lower bound for any $T \geq 1$.

2.7 Achievability

We note that many popular classical uncoded caching policies, such as LRU, LFU, and FIFO, have linear regrets (Proposition 1 of [23]). This can be simply understood from the following example: consider the single cache setting with $N = 2, C = 1$ and an alternating file request sequence $(x_t)_{t \geq 1} = \{1, 2, 1, 2, 1, 2, \ldots\}$. Since any missed content is always immediately loaded onto the cache, each of the above policies gets zero cumulative hits. On the other hand, caching either one of the files forever achieves a total of $\frac{T}{2}$ cumulative hits for a horizon of length $T$. Thus, all of the above policies have $O(T)$ regret. This is surprising as the LRU and FIFO policies are known to have a finite competitive ratio [45]. To the best of our knowledge, no uncoded caching policy with sub-linear regret is known in the literature.

Achievability with uncoded caching: Making use of the theory of Online Structured Learning [46], we now propose a simple Follow the Perturbed Leader (FTPL)-based uncoded caching policy, which achieves $O(\sqrt{T})$ expected regret against an oblivious adversary. The FTPL policy maintains a cumulative running count of the number of times a file was requested so far. This count is then perturbed by adding i.i.d. Gaussian noise of zero mean and an appropriate variance to each of the count values. Finally, at each time slot, the top $C$ files with the highest perturbed count are loaded onto the cache. The FTPL policy is formally described below in Algorithm 1.

Algorithm 1 FTPL policy for Uncoded Caching

1. $\text{count} \leftarrow 0, \eta \leftarrow \frac{1}{(4\pi \log N)^{1/4}} \sqrt{\frac{T}{C}}$
2. for $t = 1$ to $T$
3. \hspace{0.5cm} $\text{count} \leftarrow \text{count} + x_t$
4. \hspace{0.5cm} Sample $y_t \sim N(0, I_{N \times N})$
5. \hspace{0.5cm} $\text{perturbed count} \leftarrow \text{count} + \eta y_t$
6. \hspace{0.5cm} Sort $\text{perturbed count}$ in decreasing order and load the top $C$ files in the cache.
7. end for

We prove the following achievability bound for the FTPL policy.

Theorem 3 (Achievability with uncoded caching). In the single cache setting, the FTPL uncoded caching policy achieves the following upper bound for expected regret (the expectation is taken over randomness of the algorithm)

$$E_{\{y_t\}_{t \geq 1}}(R_{\text{FTPL}}^T) \leq 1.51(\log N)^{1/4} \sqrt{CT}.$$
We now formalize the above system model for caching in a CDN. A set of users \( I = \{1, 2, \ldots, I\} \) is connected to a set of caches \( J = \{1, 2, \ldots, J\} \) in the form of a bipartite network. For simplicity, we assume that the caches are homogeneous in the sense that each cache has the same storage capacity \( C \). As before, the library size (i.e., the number of all possible files) \( N \) is assumed to be sufficiently large.

The connection between the users and the caches is represented by the bipartite graph \((I, J, E)\). The set of caches connected to a user \( i \in I \) is denoted by \( \partial^+(i) \equiv \{ j \in J : (i, j) \in E \} \). Similarly, the set of users connected to a cache \( j \in J \) is denoted by \( \partial^-(j) \equiv \{ i \in I : (i, j) \in E \} \). The in-degree of the cache \( j \) is defined as \( d_j \equiv |\partial^-(j)|, j \in J \). For the sake of simplicity, we assume the network to be right \( d \)-regular, i.e., \( d_j = d, \forall j \in J \). See Figure 2 for a schematic.

Each user requests one file per time slot. Each file-request may be served by any (one or more) neighbouring caches. As before, the file request generated by a user \( i \) at time \( t \) is one-hot encoded by an \( N \)-dimensional binary vector \( x^i_t \), with the interpretation that \( x^i_{tj} = 1 \) if and only if the \( j \)-th file is requested by the user \( i \) at time \( t \). The cache configuration of the \( j \)-th cache at time \( t \) is represented by the \( N \)-dimensional vector \( y^j_t \), with each component denoting the fraction of the corresponding coded file cached. The cache configuration \( y^i_t \) must always satisfy the cache-capacity constraints. Thus, the set of all feasible cache configuration \( Y_{Jt} \) is given by:

\[
Y_{Jt} = \left\{ (y^j_t : j \in J) : \sum_{j=1}^{N} y^j_t \leq C, \forall j \in J, 0 \leq y^j \leq 1 \right\}. \tag{14}
\]

As before, for uncoded caching, we have \( y^j_{tf} \in \{0, 1\}, \forall j, f, t \).

### 3.2 Reward and Regret

For content distribution networks, it will be useful to distinguish between elastic and inelastic contents, as defined below. Making this distinction is essential due to the possibility that, in a caching network, the same content may be cached and retrieved from more than one cache at a time. Recall that, for rate less codes, it is only

the total amount of received encoded symbols that determine the decoding quality.

**Elastic contents:** We call a content to be elastic if receiving multiple layers (i.e., resolutions) of the same content improves its overall utility for the users. Examples of elastic contents include multi-bitrate video files for adaptive streaming [48], [49], multi-resolution HD videos [50], and erasure-coded files in fault-tolerant distributed file system, such as Hadoop [51], [52]. In this setting, an incoming file request \( x^i_t \) from the \( i \)-th user can be satisfied by fetching and combining parts of the cached layers of contents from different neighboring caches \( j \in \partial^+(i) \). Accordingly, for elastic contents, we define the one-slot reward to be the aggregate of the cache-hits at that slot, i.e.,

\[
q_{\text{elastic}}(x^i_t, y^j_t) = \sum_{i \in I} x^i_t \cdot \left( \sum_{j \in \partial^+(i)} y^j_t \right). \tag{15}
\]

Hence, a user’s utility increases linearly as she receives more layers of the requested content from different neighboring caches.

**Inelastic contents:** We call a content to be inelastic if the content has only a single layer (resolution) and a user is fully satisfied if she is able to retrieve the original content. Popular examples of inelastic contents include traditional webpages, databases, documents, and single resolution images. Similar to the elastic case, if the files are encoded by rate less MDS codes, a request from a user may be satisfied by combining different fractional parts of the content (from different neighboring caches), with the fractions adding up to unity. Consequently, we define the one-slot reward for inelastic content to be

\[
q_{\text{inelastic}}(x^i_t, y^j_t) \equiv \sum_{i \in I} x^i_t \cdot \min \left\{ 1, \left( \sum_{j \in \partial^+(i)} y^j_t \right) \right\}. \tag{16}
\]

where 1 is the all-one vector, and the \( \min(\cdot, \cdot) \) operator outputs a vector whose components are the pointwise minimum of the corresponding components of the input argument vectors. In comparison with the reward definition in Eqn. (15), the \( \min(1, \cdot) \) operator in Eqn. (16) takes into account the fact that receiving multiple fractions of an inelastic content, summing up to more than one, does not add additional utility. Hence, unlike the elastic contents, inelastic contents have bounded rewards. We note that the reward in Eqn. (16) coincides with the utility definition in Eqn. (13) of [23]. The regret of any caching policy is defined exactly in the same way as in the single cache case via Eqsns. (4) and (5).

### 3.3 Achievability for Caching Networks

In the following, we describe a simple and distributed gradient-based coded caching policy that achieves \( O(\sqrt{T}) \) regret in both elastic and inelastic settings. We then propose an extension of the Follow the Perturbed Leader-based uncoded caching policy given in Algorithm 1, which also achieves near-optimal regret in the elastic setting.

**Achievability with coded caching** [23]: Let \( q(x, y) \) be a generic one-slot reward function, which is concave in the cache-configuration vector \( y \) (e.g., \( q(\cdot, \cdot) \) could be chosen to be either \( q_{\text{elastic}}(\cdot, \cdot) \) or \( q_{\text{inelastic}}(\cdot, \cdot) \)). Let \( q_t \) be a supergradient of \( q(x_t, y) \) at \( y = y_t \). The

![Figure 2: Schematic of a Bipartite Caching Network](attachment:image.png)
paper [23] describes the following Online Gradient Ascent (OGA)-based caching policy: starting from any initial feasible configuration $y_0 \in Y_f$, iterate as follows:

$$y_{t+1} = \Pi_{Y_f}(y_t + \eta g_t),$$

where $Y_f$ is the set of all feasible cache configurations given in Eqn. (14), $\Pi_{Y_f}(\cdot)$ is the Euclidean projection operator on the set $Y_f$, and $\eta > 0$ is an appropriate step-size parameter. For the single user- single cache setting of Section 2, we simply set $|J| = |\mathcal{J}| = 1$.

Distributed implementation: The OGA-based caching policy can be implemented at each cache in a distributed fashion with locally available information only. This can be seen from the following two observations:

1. A separable supergradient $g_t$ for both the objective functions $q_{\text{elastic}}(\cdot)$ and $q_{\text{inelastic}}(\cdot)$ may be obtained, such that the gradient ascent steps in Eqn. (17) can be carried out locally. For an expression of such a supergradient, see Eqn. (30).
2. Since the cache capacity constraints are separable for different caches, the projection operation $\Pi_{Y_f}(\cdot)$ may be carried out separately for each cache as $\Pi_{Y_f}(g_j) = (\Pi_{y_f}(g_j))$, $j \in \mathcal{J}$, where $\mathcal{Y}$ is the single cache constraint set given by Eqn. (1).

We have the following achievable result for OGA:

**Theorem 4** (Achievability with coded caching [23]).

For both elastic and inelastic contents, the OGA-based caching policy (17) with step size $\eta = \frac{\sqrt{2T}}{d\sqrt{T}}$ achieves the following upper-bound on regret for a right $d$-regular bipartite network:

$$R_T^{\text{OGA}} \leq d|\mathcal{J}|\sqrt{2CT}.$$  

Proof outline: In this proof, we appeal to Theorem 2 of [23], which gives a generic regret upper bound for the OGA-based caching policy. We conclude the proof of Theorem 4 by computing the diameter of the feasible set $Y_f$ subject to the cache-capacity constraints. Note that we can not directly use the regret upper-bound given in Theorem 3 of [23] because there the authors make an assumption that only one user out of $|J|$ users may request for contents at a slot. In our model, there is no such restriction so that all $|J|$ users may simultaneously request for contents at a slot. Please refer to Appendix 4.4 for a proof of Theorem 4.

Achievability with uncoded caching: For uncoded caching in a bipartite network, we propose a simple extension of the FTPL policy given in Algorithm 1 for a single cache. In this extension, each of the $|\mathcal{J}|$ caches independently implements the FTPL policy irrespective of whether the content is elastic or inelastic. We have the following achievable result for elastic contents:

**Theorem 5** (Achievability with uncoded caching for elastic contents). For elastic contents, the FTPL caching policy with the noise parameter $\eta = \frac{d}{(4\pi \log N)^{1/4} \sqrt{T}}$ yields the following upper-bound on expected regret for a right $d$-regular bipartite network:

$$E_{\{Y_f\}}(R_T^{\text{FTPL}}) \leq 1.51(\log N)^{3/4}d|\mathcal{J}|\sqrt{CT}.$$  

See Appendix 7.2 for a proof of Theorem 5.

### 3.4 Converse for Caching Networks

The question of regret-optimality of the OGA policy (17) for caching networks was left open in [23] due to lack of known lower bounds. In the following, we prove tight universal lower bounds for the regret, which applies to both coded and uncoded caching.

**Theorem 6** (Lower bound for elastic contents). For caching elastic contents in a bipartite network in the above set up with $N \geq 2C$, the regret of any online caching policy $\pi$ is lower bounded as:

$$R_T^{\pi} \geq d|\mathcal{J}|\sqrt{\frac{CT}{2\pi}} - C\left(\frac{1}{\sqrt{T}}\right), \forall T \geq 1.$$  

Proof outline: In this proof, we construct a common randomized file request sequence $(X_t)_{t \geq 1}$, which is identical for each user. In other words, all users request the same random file at each slot. Thus, unlike most other applications of the probabilistic method, which usually proceeds with i.i.d. random variables, we consider a set of mutually dependent file request sequence. The expected reward accrued by any caching policy is then obtained by using the statistical symmetry of the file requests and the linearity of expectation. Finally, the static optimal caching configuration is identified, and the reward accrued by the optimal stationary policy is lower bounded by appealing to Lemma 1. Combining the above two results yields the regret lower bound. Please refer to Section 6.5 for the complete proof of Theorem 6.

The following theorem gives regret lower bound for caching inelastic content in a bipartite network.

**Theorem 7** (Lower bound for inelastic contents). For caching inelastic contents in a bipartite network in the above set up with $N \geq 2C|\mathcal{J}|$, the regret of any online caching policy $\pi$ is lower bounded as:

$$R_T^{\pi} \geq d\sqrt{\frac{|\mathcal{J}|CT}{2\pi}} - C\left(\frac{1}{\sqrt{T}}\right), \forall T \geq 1.$$  

Proof outline: The principal difficulty in extending the argument from the proof of Theorem 6 to the inelastic case is the presence of non-linearity in the reward function (16) in the form of $\min(\cdot, \cdot)$ operator. As a result, it becomes difficult to analyze the expected reward accrued by the optimal stationary caching configuration. To get around this obstacle, we lower bound the reward of the optimal caching configuration with the help of a carefully constructed sub-optimal caching configuration $Y_L$. Interestingly, under the caching configuration $Y_L$, the non-linearity of the reward function vanishes,
which leads to tractable analysis. Similar to the proof of Theorem 6, this proof also uses an identical (i.e., dependent) file request sequence across all users. Please refer to Section 6.6 for the complete proof of Theorem 7.

*Tightness:* Comparing the regret upper bounds in Theorems 4 and 5 with the lower bounds in Theorems 6 and 7, we see that, for elastic contents, the OGA and FTPL network caching policies are regret-optimal up to a constant and poly-log factors respectively. For inelastic contents, the OGA policy is regret-optimal up to a factor of $O(\sqrt{|J|})$.

## 4 EXPERIMENTS

*Dataset description:* In this section, we compare the performance of the existing caching policies with the proposed FTPL policy using a popular and stable benchmark - MovieLens 1 M dataset [37, 53]. This dataset contains ~1 M ratings for $N \approx 3700$ movies, along with the timestamps of the ratings. The ratings were given by ~6000 unique users. For our experiments, we assume that the users request movies from a CDN (such as Netflix) in the same chronological order as the recorded timestamps of the ratings. A histogram of the movie request frequencies by all users together is shown in Fig. 5 of the Appendix.

*Experimental Setup:* Following the standard industry practice, the cache capacity $C$ of each cache is set to be a fixed $\alpha$ fraction of the total library size $N$, where we take $\alpha = 0.01$. For the bipartite caching scenario, we assume that a total of $|T| = 10$ users are connected to $|J| = 4$ caches. Each cache has in-degree $d = 3$. The first cache is connected to the users 1, 2 and 3, the second cache is connected to the users 4, 5 and 6, the third cache is connected to the users 7, 8 and 9, and the fourth cache is connected to the users 1, 3 and 10. The entire dataset with ~1 M entries is uniformly divided into 10 disjoint blocks. Each user is allocated one block of the dataset. It is assumed that each user makes requests serially from its allocated block in the chronological order.

*Results:* The time-averaged regrets for different caching policies in the single cache setting and the bipartite network setting for both elastic and inelastic contents are plotted in Figure 3 (see the following page). From the plots in Figures 3 (a), 3 (b), and 3 (c), we conclude that the FTPL and LFU policies have the best performance in terms of average regret uniformly in all cases, and there is hardly any noticeable difference between their performance for large enough $T$. These two policies also perform very close to the theoretical lower bounds (corresponding to the worst-case request sequence). On the other hand, we find that the LRU policy has the worst performance, followed by OGA, which performs only marginally better. It is quite surprising to find that the uncoded caching policy FTPL outperforms the coded caching policy OGA in all scenarios. Figure 6 in the Appendix shows the variation of the average regret as the cache capacity is increased from 1 percent to 10 percent of the library size for a fixed time $T = 5 \times 10^4$. These plots confirm that the regret increases with the cache size. However, we find that the increase in the regret is smallest for the LFU and FTPL policies followed by the OGA and the LRU policy.

## 5 CONCLUSION AND FUTURE WORK

In this paper, we obtain tight sub-linear regret lower bounds for the online caching problem for single caches and bipartite caching networks. In the process, we derive a key technical result on the balls-into-bins problem and utilize the result in deriving all our lower bounds. We also propose a new randomized caching policy, called FTPL, which is shown to be both sound in theory and superior in practice. We envision the following future research directions stemming from this work: (1) As an immediate follow-up, it will be interesting to narrow down the $O(\sqrt{|J|})$ gap between the lower and upper regret bounds for inelastic contents in a bipartite caching network. Moreover, obtaining a regret guarantee for the FTPL policy for bipartite networks with inelastic contents would be nice. (2) We defined the reward functions primarily with the online performance of the caching policies (i.e., hit rates) in mind. In particular, our reward definitions do not take into account the system cost associated with cache replacements at every slot. Hence, it would be interesting to design an uncoded caching policy, which makes incremental changes to the caching configuration at every slot, yet matches the sublinear regret lower bounds. (3) A variation of the caching problem arises in the context of inventory management where, instead of digital files, physical commodities are stored in the caches (e.g., retail stores). Requests for the commodities arrive sequentially. The requested commodities, which are currently present in the cache, are immediately removed from the cache (e.g., sold). Hence, unlike in our setting, there is no scope of "coding", and it would make sense to cache multiple copies of the same commodity at the same slot, subject to the cache capacity constraints (c.f., Eqn. (1)). The performance of FTPL-like randomized caching policies will be exciting to investigate in this setup. (4) Finally, it would also be interesting to go beyond the single-hop setting of bipartite caching networks and design a regret-optimal joint routing and caching policy for multi-hop CDNs [54].

## 6 PROOF OF THE RESULTS

### 6.1 Proof of Lemma 1

We index the bins sequentially as 1, 2, ..., 2$C$. Next, we logically combine every two consecutive bins $\{2i-1, 2i\}$, $1 \leq i \leq C$, to obtain $C$ Super bins (See Figure 4). Let us denote the (random) number of balls in the $i$th super bin by $X_i$; $i = 1, 2, \ldots, C$. Conditioned on the r.v. $X_i$, the number of balls in the corresponding bins $2i-1$ and $2i$ are jointly distributed as $(Z, X_i - Z)$, where $Z$ is a binomial random variable with parameter $(X_i, \frac{1}{2})$. Let $H_i$ denote the maximum number of balls between the corresponding bins $2i-1$ and $2i$. Then, as shown in the proof Theorem 1, when $X_i > 0$:

$$
\mathbb{E}(H_i | X_i) \geq \frac{X_i}{2} + \sqrt{\frac{X_i}{2\pi} - \frac{1}{2\sqrt{2\pi X_i}}}, \ \forall 1 \leq i \leq C.
$$ (18)
Now, observe that if \( X_1 = 0 \) then \( H_1 = 0 \) a.s. Hence, almost surely, we have \( H_1 = H_1 \mathbb{1}(X_1 > 0) \). The equation (c) follows from the tower property of conditional expectation, the inequality (d) follows from the bound (18), and the equality (e) follows from the facts that \( X_1 = X_1 \mathbb{1}(X_1 > 0) \), \( \sqrt{X_1} = \sqrt{X_1} \mathbb{1}(X_1 > 0) \). The Lemma now follows by using the bounds on moments of the binomial distribution as computed next.

Bounding the Expectations in Eqn. (19): For bounding the middle term in Eqn. (19), consider the factorization [55]:

\[
\sqrt{x} - \left(1 + \frac{x - 1}{2} - \frac{(x - 1)^2}{2}\right) = \frac{\sqrt{x}}{2}(\sqrt{x} - 1)^2(\sqrt{x} + 2).
\]

The RHS is non-negative for any \( x \geq 0 \) Thus, we have the following algebraic inequality:

\[
\sqrt{x} \geq 1 + \frac{x - 1}{2} - \frac{(x - 1)^2}{2}, \quad \forall x \geq 0.
\]

Replacing the variable \( x \) with the random variable \( \frac{X_1}{\mathbb{E}(X_1)} \) point wise, we have almost surely,

\[
\sqrt{X_1} \geq \frac{X_1}{\mathbb{E}(X_1)} = 1 + \frac{X_1}{\mathbb{E}(X_1)} - \frac{(X_1)^2}{2}\mathbb{E}(X_1)^2.
\]

Taking expectation of both sides, the above yields

\[
\mathbb{E}(\sqrt{X_1}) \geq \sqrt{\mathbb{E}(X_1)} \left(1 - \frac{\text{Var}(X_1)}{2\mathbb{E}(X_1)^2}\right), \quad (20)
\]

Finally, recall that \( X_1 \sim \text{Binom}(T, \frac{1}{T}) \). Hence, \( \mathbb{E}(X_1) = \frac{T}{C} \) and \( \text{Var}(X_1) = T \frac{1}{C} (1 - \frac{1}{C}) \leq \frac{T}{C} \). Using this, Eqn. (20) yields the following lower bound

\[
\mathbb{E}(\sqrt{X_1}) \geq \sqrt{\frac{T}{C}} - \frac{1}{2} \frac{\sqrt{C}}{T}. \quad (21)
\]

For bounding the last term in Eqn. (19), we write

\[
\frac{\mathbb{I}(X_1 > 0)}{\sqrt{X_1}} \leq 1 \mathbb{I}(X_1 \leq \frac{T}{2C}) + \sqrt{\frac{2C}{T}}.
\]

Recall that \( X_1 \sim \text{Binom}(T, \frac{1}{T}) \). Taking expectation of both sides of the above inequality, we have

\[
\mathbb{E} \left( \frac{\mathbb{I}(X_1 > 0)}{\sqrt{X_1}} \right) \leq \mathbb{E}(X_1) \left(\frac{1}{2} \frac{\mathbb{E}(X_1)}{\sqrt{X_1}}\right) + \sqrt{\frac{2C}{T}}.
\]
The probability term in the above expression may be bounded using Chebyshev’s inequality as follows:

\[ \mathbb{P}(X_t > \frac{1}{2} E(X_t)) \leq \mathbb{P}(X_t - E(X_t) > \frac{1}{2} E(X_t)) \]
\[ \leq \mathbb{P}(|X_t - E(X_t)| > \frac{1}{2} E(X_t)) \]
\[ \leq \frac{4 \text{Var}(X_t)}{(E(X_t))^2} = \frac{4C - 1}{T} \]

Taking the above bounds together, we obtain

\[ \mathbb{E} \left( \frac{1(X_t > 0)}{\sqrt{X_t}} \right) \leq \sqrt{\frac{2C}{T}} + \frac{4C}{T}. \quad (22) \]

Finally, combining Eqsns. (19), (21), and (22) together, we obtain

\[ \mathbb{E}(M_C) \geq \frac{T}{2} + \sqrt{\frac{C T}{2π}} - \left( \frac{\sqrt{2} + 4}{2\sqrt{2πT}} \right) \]

6.3 Proof of Theorem 3

Our proof follows a similar line of arguments as the proof of Theorem 1 of [46]. However, we improve the regret upper bound by a factor of \( O(\sqrt{C}) \). This additional improvement results from making use of the constraint that only one file is requested at every slot. The notations of [46] are slightly altered in order to remain consistent throughout the paper.

Let the set \( \mathcal{Y} \) denote the set of all possible uncoded caching configuration in the single cache setting. Clearly, \( |\mathcal{Y}| = \binom{N}{C} \).

Define the potential function

\[ \Phi_\eta(x) = \mathbb{E}_{y \sim \mathcal{N}(0, I)} \left[ \max_{y \in \mathcal{Y}} \langle y, x + \eta y \rangle \right]. \]

Also, denote the cumulative file request arrivals to the cache up to time \( t - 1 \) by \( X_t = \sum_{t=0}^{t-1} X_t \), with \( X_0 = 0 \). Then, as shown in Eqn. (3) of [46], the expected regret of the FTPL policy in Algorithm 1 with noise variance \( \eta^2 \) is upper bounded as \(^5\)

\[ \mathbb{E}(R_T) \leq \Phi_\eta(X_1) + \frac{1}{2} \sum_{t=1}^{T} \sum_{i=1}^{N} \langle x_t, \nabla^2 \Phi_\eta(x_t)x_t \rangle. \]

(25)

for some \( x_t \), connecting the line segment \( X_t \) and \( X_{t+1} \).

Next, we bound each of the above two terms separately. The first term may be bounded in the same way as in [46]:

\[ \Phi_\eta(X_1) \leq \eta \sqrt{2C \log \left( \binom{N}{C} \right)} \leq \eta C \sqrt{2 \log N}, \]

where the last inequality follows from the fact that \( \binom{N}{C} \leq N^C \).

Since only one file is requested at every slot, the quadratic form above may be upper bounded as

\[ \langle x_t, \nabla^2 \Phi_\eta(x_t)x_t \rangle \leq \max_{i,x} \langle \nabla^2 \Phi_\eta(x) \rangle_{ii}. \]

(26)

Moreover, following Lemma 7 of [56], we have that

\[ \langle \nabla^2 \Phi_\eta(x) \rangle_{ii} \leq \frac{1}{\eta} \mathbb{E} \left[ |\hat{g}(x_t + \eta y)|^2 \right]. \]

where \( \hat{g}(z) \in \arg \max_{y \in \mathcal{Y}} \langle y, z \rangle \). Hence, using Jensen’s inequality we have that

\[ \langle \nabla^2 \Phi_\eta(x) \rangle_{ii} \leq \frac{1}{\eta} \mathbb{E} \left[ |\hat{g}(x_t + \eta y)|^2 \right] \leq \frac{1}{\eta} \mathbb{E} \left[ |y|^2 \right] \leq \frac{1}{\eta} \sqrt{\frac{2}{\pi}}. \]

(27)

where the inequality (a) follows from the fact that for all \( y \in \mathcal{Y} \), we have \( y_t \in [0, 1] \), and the equality (b) follows from the fact that \( y_t \sim N(0, 1) \). Hence, substituting the above bounds in Eqn. (25), we

5The signs are flipped as we are in the rewards maximization setting, as opposed to the loss minimization setting of [46].
have the following upper bound on expected regret under the FTPL caching policy:

$$\mathbb{E}(R_T) \leq C\sqrt{2\log N + \frac{T}{\eta\sqrt{2\pi}}}.$$ 

Finally, choosing $\eta = \frac{1}{(4\sqrt{\log N})^4}\sqrt{\frac{T}{\pi}}$, yields the following regret upper bound for the FTPL policy:

$$\mathbb{E}(\gamma_i)_{\text{FTPL}} \leq 1.51(\log N)^{1/4}\sqrt{CT}.$$ 

### 6.4 Proof of Theorem 4

From Eqn. (15), we know that $q_{\text{elastic}}(x, y)$ is linear (and hence, concave) in the cache-configuration vector $y$. Moreover, since the pointwise minimum of linear functions is concave [57], it follows from Eqn. (16) that the reward function $q_{\text{elastic}}(x, y)$ is also concave in the cache-configuration vector $y$. To obtain a regret upper bound for the the OGA algorithm [17], we appeal to Theorem 2 of [23], which states that with an appropriate choice of the step-size parameter $\eta$,

$$R_{\text{OGA}}^T \leq \text{diam}(Y_T)L\sqrt{T},$$

where $\text{diam}(Y_T)$ denotes the Euclidean diameter [58] of the set $Y_T$ defined in (14), and $L$ is an upper-bound for the 2-norm of the (super) gradient of the reward function.

For bounding the diameter, consider any two vectors $y$ and $z$ from the set $Y_T$. We have

$$\|y - z\|_2^2 = \sum_{j \in J} \sum_{f=1}^N (y_j^f - z_j^f)^2 \leq (a) \sum_{j \in J} \sum_{f=1}^N |y_j^f - z_j^f|^2 \leq (b) \sum_{j \in J} \sum_{f=1}^N \sum_{i \in \mathcal{A}^{\text{elastic}}(j)}^N z_i^f \leq 2CJ,$$

where the inequality (a) follows from the fact that $|y_j^f - z_j^f| \leq 1$, the inequality (b) follows from triangle inequality, and finally, inequality (c) follows from the cache-capacity constraints. Since, the above bound is valid for any two vectors in the set $Y_T$, it follows that

$$\text{diam}(Y_T) \equiv \sup_{y, z \in Y_T} \|y - z\| \leq \sqrt{2CJ}.$$ 

Next, we bound the norm of the supergradients of the reward functions. Recall,

$$q_{\text{elastic}}(x, y) = \sum_{i \in I} x_i \cdot \left( \sum_{j \in \mathcal{J}} y_j^i \right).$$

Hence, we have the following expression for the supergradient of the objective function:

$$\left\langle \nabla_y q_{\text{elastic}}(x, y) \right\rangle_f^j = \sum_{i \in \mathcal{A}^{\text{elastic}}(j)} x_i^f.$$ 

Thus,

$$\|\nabla_y q_{\text{elastic}}(x, y)\|_2^2 = \sum_{j \in J} \sum_{f=1}^N \left( \sum_{i \in \mathcal{A}^{\text{elastic}}(j)} x_i^f \right)^2.$$

On the other hand, since $x \geq 0$, it follows that the vector $\nabla_y q_{\text{elastic}}(x, y)$ can be taken to be a supergradient of the concave function $q_{\text{elastic}}(x, y)$ w.r.t. the argument $y$. Thus, to obtain the regret upper bound, it only remains to upper bound the RHS of Eqn. (31). We have:

$$\left( \sum_{i \in \mathcal{A}_T} x_i^{T} \right)^2 \leq d \sum_{i \in \mathcal{A}_T} (x_i^{T})^2 \leq d \sum_{i \in \mathcal{A}_T} x_i^{T},$$

where the inequality (a) follows from Cauchy-Schwartz inequality and (b) follows from the fact that each $x_i^f$’s are either zero or one.

Hence, the RHS of Eqn. (31) is upper bounded as follows:

$$\sum_{j \in J} \sum_{f=1}^N \left( \sum_{i \in \mathcal{A}^{\text{elastic}}(j)} x_i^f \right)^2 \leq d \sum_{j \in J} \sum_{f=1}^N \sum_{i \in \mathcal{A}^{\text{elastic}}(j)} x_i^f \leq d \sum_{j \in J} \sum_{f=1}^N \sum_{i \in \mathcal{A}^{\text{elastic}}(j)} x_i^f \leq d^2 |J|,$$

where (a) follows from the facts that each user can request at most one file per slot, and the network is right-regular. Hence, for both elastic and inelastic reward functions, the 2-norm of the supergradients is upper bounded as

$$L \leq d \sqrt{|J|}.$$ 

Hence, combining Eqns. (28), (29), and (32), we have that

$$R_{\text{OGA}}^T \leq d|J|\sqrt{2CT}.$$ 

### 6.5 Proof of Theorem 6

We prove a slightly general result without assuming the bipartite caching network to be right-regular. Let $x_i^f$ be the file request vector from the $i^{th}$ user, and $y_i^j$ be the cache configuration vector of the $j^{th}$ cache selected by the policy $\pi$ at time $t$, where $i \in I, j \in J$. Hence, due to the elastic nature of the content, the total reward $G_T^\pi$ accrued by the caching policy $\pi$ is given by Eqn. (15):

$$G_T^\pi = \sum_{i=1}^T \sum_{j \in \mathcal{J}} x_i^f \cdot \sum_{j \in \mathcal{J}} y_i^j.$$ 

Recall that, while computing the regret of a policy $\pi$ via Eqn. (4), we compare the reward accrued by the policy $\pi$ to that of a stationary caching policy $\pi^*$ equipped with the hindsight knowledge (i.e., $\pi^*$ knows the entire file request sequence $\{x_t^f\}_t$ in advance, viz., Eqns. (4)-(5)). Let $y_i^j$ be the optimal stationary cache configuration vector at the server $j \in J$ set by the policy $\pi^*$. Then the reward accrued by the stationary caching policy $\pi^*$ is given by:

$$G_T^{\pi^*} = \sum_{i=1}^T \sum_{j \in \mathcal{J}} x_i^f \cdot \sum_{j \in \mathcal{J}} y_i^j.$$ 

Rearranging the terms, we have

$$G_T^{\pi^*} = \sum_{j \in J} \sum_{i \in I} x_i^f \cdot \left( \sum_{j \in \mathcal{J}} \sum_{i=1}^T x_i^f \right).$$ 

Since the $j^{th}$ cache has a total capacity of $C$, from the above equation, it is clear that the optimal stationary configuration $y_i^j$ corresponds to caching the most popular $C$ files requested by all of $j$’s
in-neighbors taken together. Following the probabilistic method described in Section 2.4, we now construct a randomized file request sequence which are identical for each user, i.e., all users request the same random file at every time slot. At time slot $t$, the file request from the $i^{th}$ user is given by the vector $X_i = X_t$, where the vector $X_t$ is sampled independently and uniformly at random from the set of first $2C$ unit vectors ($e_i \in \mathbb{R}^N, 1 \leq i \leq 2C$). With this setup, the expected reward accrued by the policy $\pi$ (i.e., the second term in Eqn. (4)) may be evaluated as:

$$E \left( \sum_{t=1}^{T} \sum_{i \in I} X_i^j \cdot Y_i^j \right) = \sum_{t=1}^{T} \sum_{i \in I} \sum_{j \in \partial^*(i)} \sum_{f=1}^{N} E(X_{tf}) Y_i^j$$

$$\leq \frac{1}{2C} \sum_{t=1}^{T} \sum_{i \in I} \sum_{j \in \partial^*(i)} \sum_{f=1}^{N} Y_i^j$$

where the equation (a) uses the fact that the random variables $X_t$ and $Y_i$ are independent and $E(X_{tf}) = \frac{1}{2C}, \forall t, f$. The inequality (b) follows from the fact that each cache has capacity $C$, and the equality (c) follows from interchanging the order of the summations.

From Eqn. (33), we can express the expected total reward $G_T^\pi$ accrued by the stationary policy $\pi^*$ as:

$$E(G_T^\pi) = \sum_{j \in J} \sum_{n=0}^{N} E(M_n^j),$$

where $M_n^j$ denotes the sum of the largest $N$ coordinates of the following $N$-dimensional random vector:

$$\sum_{t=1}^{T} \sum_{i \in \partial^*(j)} X_i^j = d_j \sum_{t=1}^{T} X_t.$$

Thus, it follows that the random variable $M_n^j$ is statistically identical to $d_j$ times the total number of balls in the most occupied $C$ bins when $T$ number of balls are thrown independently and uniformly at random into $2C$ bins. Finally, appealing to Lemma 1, we have

$$E(M_n^j) \geq d_j \left( \frac{T}{2} + \sqrt{\frac{CT}{2\pi}} - O(\frac{1}{\sqrt{T}}) \right).$$

(34)

Finally, combining the above, we have the following regret lower bound for caching elastic contents in a bipartite caching network:

$$R_T^\pi \geq \sqrt{\frac{CT}{2\pi}} \left( \sum_{j \in J} d_j \right) - O(\frac{1}{\sqrt{T}}) = d|J| \sqrt{\frac{CT}{2\pi}} - O(\frac{1}{\sqrt{T}}).$$

6.6 Proof of Theorem 7

We begin our analysis with the following two observations:

1. From the definition of the reward functions (15) and (16), in general, we have,

$$q_{\text{inelastic}}(x, y) \leq q_{\text{elastic}}(x, y), \forall x, y.$$

(35)

2. In the special case, when different servers $j \in J$ cache different items, i.e., $y^j \cdot y^{j'} = 0, \forall i \neq j$, we have

$$q_{\text{inelastic}}(x, y) = q_{\text{elastic}}(x, y), \forall x, y.$$

(36)

Eqn. (36) follows from the fact that in the product $x_i^j \cdot (\sum_{j \in \partial^*(i)} y_j^j) = \sum_{j \in \partial^*(i)} x_i^j \cdot y_j^j$, only one of the dot-product terms could be strictly positive, as different servers cache different items and the vector $x_i^j$ has only one positive component. Moreover, the value of each cod product is at most one, as each server stores at most one copy of each file.

With these observations at hand, we now relate our argument below to the proof of Theorem 6 to obtain a regret lower bound for inelastic contents. Similar to the proof of Theorem 6, we construct a randomized and identical file request sequence $(X_i^j, X_t)_{t \geq 1}$ for each user, where the random variable $X_t$ is sampled independently and uniformly at random from the set of first $2C|J|$ unit vectors $(e_i \in \mathbb{R}^N, 1 \leq i \leq 2C|J|)$. Hence, using observation (1) and working similarly as in Eqn. (34), the expected total reward $G_T^\pi$ accrued by any online caching policy may be upper bounded as

$$G_T^\pi \leq \sum_{j \in J} \sum_{n=0}^{N} \sum_{t=1}^{T} X_i^j \cdot Y_i^j \leq \frac{T}{2|J|} \sum_{j \in J} d_j = \frac{Td}{2}. \quad (37)$$

(37)

Note that, unlike the elastic setting, due to the presence of the min operator in the reward function (16), obtaining an optimal cache configuration vector $Y^*$ is non-trivial in the inelastic setting. However, since we only require a lower bound to the total expected reward accrued by the optimal static policy $\pi^*$, any suitably constructed sub-optimal caching configuration will serve the purpose, provided we can evaluate its reward. Towards this end, in the following, we construct an $N|J|$ dimensional stationary cache-configuration vector $Y_L$ (obtained by stacking the $N$-dimensional cache-configuration vectors $Y_i^j$ together $\forall j \in J$) with the property that

$$Y_i^j \cdot Y_i^j = 0, \forall i \neq j, i, j \in J.$$

(38)

(38)

The cache-configuration vector $Y_L$ is constructed by first sorting the vector $\upsilon \equiv \sum_{t=1}^{T} X_t$ in non-increasing order. Then, we let the cache configuration $Y_L^j$ correspond to the set of $C$ files from rank $(j-1)C+1$ to $jC$. Clearly, the above construction ensures property (38). Let the operator $S_{\upsilon}(m, n)$ denote the sum of the coordinates running from $m$ to $n$ of the vector $\upsilon$, sorted in non-increasing order. With the above construction, since all users make the same file request at each time ($i.e., X_i^j = X_t, \forall i$), we have $\forall j \in J$:

$$Y_i^j \cdot \left( \sum_{t=1}^{T} X_t^j \right) = dY_i^j \cdot \upsilon$$

$$= dS_{\upsilon}(j-1)C+1, jC).$$

(39)
Finally, the reward accrued by the optimal stationary policy $\pi^*$ may be lower-bounded as:

$$
G^\pi_T \geq (a) \mathbb{E}\left( \sum_{j \in J} Y^j_1 \cdot \left( \sum_{i \in \partial_j} \sum_{t=1}^T X^i_t \right) \right)
$$

$$
= (b) \sum_{j \in J} \mathbb{E}\left( Y^j_1 \cdot \left( \sum_{i \in \partial_j} \sum_{t=1}^T X^i_t \right) \right)
$$

$$
= (c) d \sum_{j \in J} \mathbb{E}\left( S_{\pi}(j-1)C + 1, jC \right)
$$

$$
= d \mathbb{E}\left( \sum_{j \in J} S_{\pi}(j-1)C + 1, jC \right)
$$

$$
= d \mathbb{E}\left[ S_{\pi}(1, |J|C) \right]
$$

$$
\geq \frac{Td}{2} + d \sqrt{\frac{|J|C T}{2\pi}} - O\left( \frac{1}{\sqrt{T}} \right), \quad (40)
$$

where the inequality (a) follows from Eqn. (36) of observation 2, the equation (b) follows from the linearity of expectation, and the inequality (c) follows from Eqn. (39). Finally, we realize that the random variable $S_{\pi}(1, |J|C)$ is distributed as the total load of the most loaded $|J|C$ bins when a total of $T$ balls are thrown uniformly at random to $2|J|C$ bins. Hence, the inequality (d) follows from an application of Lemma 1. Finally, combining Eqns. (37) and (40), we have

$$
R^\pi_T \geq G^\pi_T - G^\pi_T \geq d \sqrt{\frac{|J|C T}{2\pi}} - O\left( \frac{1}{\sqrt{T}} \right). \quad \blacksquare
$$
We recall Robbin’s form of Stirling’s formula \[44\], which will be useful in obtaining a non-asymptotic bound for regret.

\[
\sqrt{2\pi n^{\frac{3}{2}} e^{-n} e^{\pi n}} \leq n! \leq \sqrt{2\pi n^{\frac{3}{2}} e^{-n} e^{\pi n}}. \tag{41}
\]

Consider the following two possible cases.

**Case I: T is even**

In this case, we lower bound the Mean absolute deviation in Eqn. (11) as follows:

\[
\mathbb{E}[|Z - \frac{T}{2}|] = \frac{2}{2^T+1} \left(\frac{T}{2} + 1\right) \left(\frac{T}{2} + 1\right)
\]

\[
= \frac{T!}{2^T+1} \left(\frac{T}{2} + 1\right)
\]

\[
\geq \frac{1}{2^T \sqrt{2\pi}} \left(\frac{T}{2} + 1\right)^{\frac{T}{2} + 1} e^{-\frac{T}{2} - \frac{1}{12\pi}}
\]

\[
= \frac{\sqrt{T}}{2\pi} - \frac{\pi}{3\sqrt{2\pi T}}
\]

where the inequality (a) follows from the bound (41) and the inequality (b) follows from the fact that \(e^{-x} \geq (1 - x)\) for all \(x \in \mathbb{R}\). Using the fact that \(\frac{9T + 1}{12T + 1} < 1\) for all \(T > 0\), the mean deviation can be lower bounded as:

\[
\mathbb{E}[|Z - \frac{T}{2}|] \geq \frac{\sqrt{T}}{2\pi} - \frac{1}{3\sqrt{2\pi T}} \tag{42}
\]

**Case II: T is odd**

In this case, the binomial coefficient \(\left(\frac{T}{2}\right)_{+1}\) may be lower bounded as follows:

\[
\left(\frac{T}{2}\right)_{+1} = \frac{T^2}{(\frac{T}{2})^2!}
\]

\[
\geq \frac{1}{\sqrt{2\pi}} \left(\frac{T}{2} + 1\right)^{\frac{T}{2} + 1} e^{-\frac{T}{2} - \frac{1}{12\pi}}
\]

\[
\frac{T^{T + \frac{1}{2}}}{(T - 1)^{\frac{T}{2}}} \geq \frac{T^{T + \frac{1}{2}}}{(T - 1)^{\frac{T}{2}}} = 2T^{\sqrt{T}}.
\]

Also, for \(T \geq 3\)

\[
\frac{1}{6} + \frac{6}{T} + \frac{6}{T - 12 + \frac{T}{2}} \leq \frac{1}{2}.
\]

This gives us the following lower bound

\[
\left(\frac{T}{2} + 1\right) \geq \frac{1}{\sqrt{2\pi}} \frac{2^{T + 1} + \sqrt{T}}{T + 1} \left(1 - \frac{1}{2T}\right).
\]

For \(T = 1\), the inequality holds good by direct inspection. Hence, the mean deviation from Eqn. (11) is lower bounded as

\[
\mathbb{E}|Z - T| \geq \frac{1}{2\pi} \frac{T + 1}{2} \sqrt{\frac{T + 1}{4\pi}} \left(1 - \frac{1}{2T}\right)
\]

\[
\geq \frac{T}{2\pi} - \frac{1}{2\sqrt{2\pi T}} \tag{43}
\]

Finally, combining equations (42) and (43) together, we have the following lower bound for all \(T \geq 1\)

\[
\mathbb{E}|Z - T| \geq \frac{T}{2\pi} - \frac{1}{2\sqrt{2\pi T}}
\]

which verifies Eqn. (12).

### 7.2 Proof of Theorem 5

Let \(R^\text{single}_T\) denote the regret of a single cache under the FTPL caching policy. Due to linearity of the reward functions for elastic contents, the regret for bipartite network caching may be simply upper bounded by summing the regret for each of the \(|J|\) constituent single caches, i.e.,

\[
R_T \leq |J| R^\text{single}_T. \tag{44}
\]

However, note that we cannot use the regret upper bound from Theorem 3 to upper bound the regret of the individual caches, as, unlike the setting of Theorem 3, each individual cache may receive up to \(d\) requests from its neighboring users per slot. Hence, we need to modify the proof of Theorem 3 to take into account. Following the same line of argument as in the proof of Theorem 6.3, the inequality (26) may be replaced by the following

\[
(x_i, \nabla^2 \Phi_\eta(x_i) x_i) \leq d^2 \max_{i,j} (|\nabla^2 \Phi_\eta(x)|)_{ij}. \tag{45}
\]

Eqn. (45) follows from the fact that since the number of non-zero terms in \(x_i\) is at most \(d\) (i.e., \(|x_i|_0 \leq d\), \(|x_i|_1 \leq d\)). Finally, proceeding similarly as in Eqn. (27), we conclude that

\[
(|\nabla^2 \Phi_\eta(x)|)_{ij} \leq \frac{1}{\eta} \frac{\sqrt{T}}{\sqrt{\pi}} \forall x, i, j.
\]

Plugging in the above in Eqn. (25) results in the following regret bound for the FTPL caching policy:

\[
\mathbb{E}(R^\text{single}_T) \leq C\sqrt{\frac{\log N}{\eta}} + d^2 \frac{T}{\eta \sqrt{2\pi}}. \tag{46}
\]

Choosing \(\eta = \frac{d}{(4\pi \log N)^{1/4}}\sqrt{\frac{T}{C}}\), and combining Eqns (44) and (46), we have the following regret upper bound for the FTPL policy for caching elastic contents in a \(d\)-right regular bipartite caching network:

\[
\mathbb{E} \{\gamma_T\}_e (R_T) \leq 1.51 (\log N)^{1/4} d |J| \sqrt{T}.
\]
7.3 Additional Plots

In this Section, we include two additional experimental results above.