LeadCache: Regret-Optimal Caching in Networks

Abhishek Sinha

Assistant Professor of EE

IIT Madras

Debjit Paria

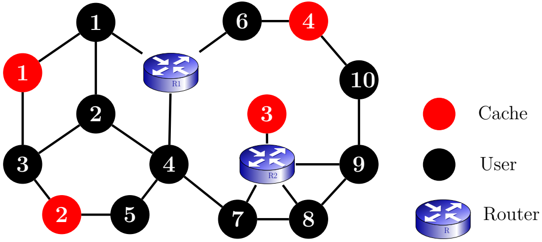
Chennai Mathematical Institute

NeurIPS 21

Tuesday 12^{th} October, 2021

The Network Caching Problem

Consider the problem of retrieving movies from the Netflix servers



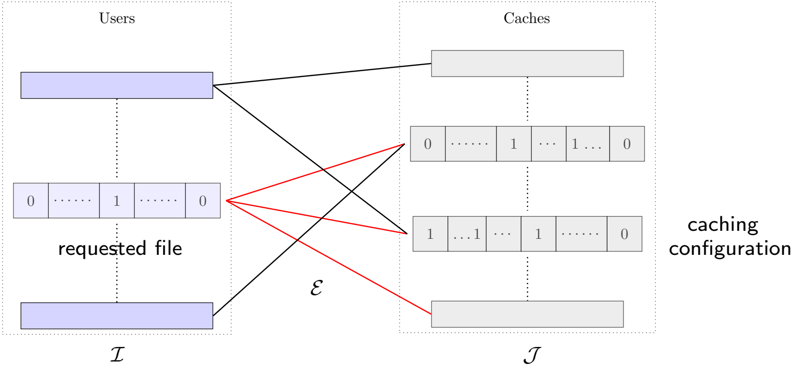
Schematic of a Content Distribution Network (CDN)

Problem: How to predict the future file requests and cache the files optimally across thousands of servers distributed across the globe?

History and Related Work

- The caching (*a.k.a. paging*) problem has been studied for more than sixty years
- Two distinct lines of work:
  - Adversarial requests: minimizes the Competitive Ratio.
  - Stochastic requests: maximizes the hit-rate (*e.g.*, with Zipf's popularity distribution). Recently, there has been a surge of activities on coded caching as well.
- Due to the complex interactions among the caches, the majority of the works on network caching assume a stochastic model
  - Negative result in the adversarial setting: Unbounded competitive ratio for deterministic algorithms (Vaze et al. 2016)
- With volatile content popularity, the stationarity assumption does not hold in practice
  - Need a learning-based policy that can learn the transient file-request patterns on-the-fly
- Our contribution: Uncoded network caching to minimize regret using tools from online learning theory

Bipartite Caching

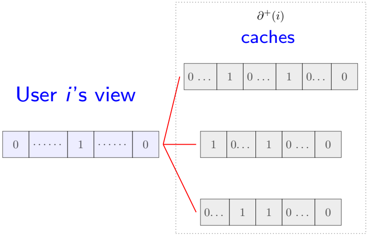


Bipartite-network

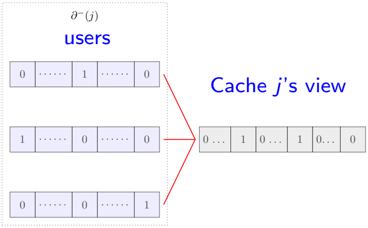
- Network given by a Bipartite Graph $\mathcal{G}$
- User i is connected to cache j iff it can retrieve a file from cache j in the original network
- Each cache has a limited storage capacity of C files

Notations

- Cache Configuration: $\boldsymbol{y}_t^j \in \{0,1\}^N$ denotes the set of files in cache $\boldsymbol{j}$ at time $\boldsymbol{t}$, with $\sum_{f=1}^N \boldsymbol{y}_{t,f}^j \leq C$.
- File Requests: $\boldsymbol{x}_t^i \in \{0,1\}^N$ is the requested file by user $\boldsymbol{i}$ at time $\boldsymbol{t}$, with $\sum_{f=1}^N \boldsymbol{x}_{t,f}^i = 1$.



$$\boldsymbol{i} \in \mathcal{I},\ \partial^+(\boldsymbol{i}) = \{\boldsymbol{j} : \boldsymbol{j} \in \mathcal{J}, (\boldsymbol{i},\boldsymbol{j}) \in E\}$$



$$\boldsymbol{j} \in \mathcal{J},\ \partial^-(\boldsymbol{J}) = \{\boldsymbol{i} : \boldsymbol{i} \in \mathcal{I}, (\boldsymbol{i},\boldsymbol{j}) \in E\}$$

Problem Statement

- A user receives a cache-hit if the requested file is currently stored in at least any one of the caches connected to the user.
- The reward at slot t is given by the total number of cache-hits:

$$q(\boldsymbol{x}_t, \boldsymbol{y}_t) \equiv \sum_{i \in \mathcal{I}, f \in [N]} x_f^i(t) \cdot \left(\min \left(1, \sum_{j \in \partial^+(i)} y_f^j\right)\right).$$

- The user requests are decided by the adversary and the online policy decides the cache configuration before the requests arrive at each slot.

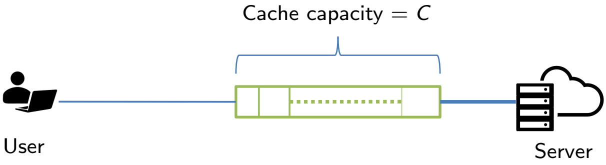
- Performance metrics: (1) Hit rate (measured by Static regret):

$$\mathbb{E}\big(R^\pi(T)\big) \stackrel{\text{(def.)}}{=} \mathbb{E}\Bigg[\sup_{\{\boldsymbol{x}_t\}_{t=1}^T} \left(\sum_{t=1}^T q(\boldsymbol{x}_t, \boldsymbol{y}^*) - \sum_{t=1}^T q(\boldsymbol{x}_t, \boldsymbol{y}_t^\pi) \right)\Bigg]$$

where $\boldsymbol{y}^* = \arg\max_{\boldsymbol{y} \in \mathcal{Y}} \sum_{t=1}^T q(\boldsymbol{x}_t, \boldsymbol{y})$, the best fixed offline configuration

(2) Cache refresh rate - need to minimize to avoid network congestion

Warm up: Single Cache results [Bhattacharjee et al. 2020]



Caching with a single cache

- Lower bound: For any $N \geq 2C$, the regret of any online caching policy π is lower bounded as

$$R_T^\pi \geq \sqrt{\frac{CT}{2\pi}} - \Theta(\frac{1}{\sqrt{T}}).$$

- Upper-Bound: A Follow-the-Perturbed Leader (FTPL)-based caching policy (described next) achieves

$$\mathbb{E}(R_T^\pi) \leq 1.51(\log(N/C))^{1/4}\sqrt{CT}.$$

Single Cache Policy

- As the requests are adversarial, the greedy strategy of storing the C most frequently requested files does not work.
- Surprisingly, the above strategy works if we add independent noise to the frequency counts!

Follow The Perturbed Leader for single cache

Add scaled standard Gaussian noise to the cumulative file count $\boldsymbol{X}(t)$, and then cache the top C files, i.e.,

$$\boldsymbol{y}_t = \operatorname*{argmax}_{c \in \mathcal{C}} \langle \boldsymbol{\Theta}(t), c \rangle$$

where $\Theta(t) = \boldsymbol{X}(t) + \eta\gamma$ and $\gamma \overset{\text{i.i.d.}}{\sim} \mathcal{N}(\mathbf{0}, \mathbf{1}_{N\times 1}), \eta = O(\sqrt{T})$.

Observations:

- The FTPL policy fetches at most one file at a slot, and hence, is bandwidth efficient.
- Popular caching policies, such as LRU, LFU, FIFO, MARKER are provably sub-optimal as they all have linear regret.

Design of The LeadCache Policy

- The main obstacle in extending the previous FTPL policy is the *non-linearity* of the reward function.
- To get around this issue, we switch to a *virtual action domain* $\mathcal{Z}$

$$q(\boldsymbol{x}_t, \boldsymbol{y}_t) \equiv \sum_{i \in \mathcal{I}} \boldsymbol{x}^i(t) \cdot \underbrace{\left(\min \Big(\mathbf{1}, \sum_{j \in \partial^+(i)} \boldsymbol{y}^j \Big) \right)}_{\geq \boldsymbol{z}(t)}$$

- At every step we need to "*solve*" an Integer Linear Program (more about this later) and then apply FTPL:

$$\boldsymbol{z}_t \in \arg\max_{z \in \mathcal{Z}} \left(\sum_{i \in \mathcal{I}} \boldsymbol{\Theta}_i(t) \cdot \boldsymbol{z}_i \right),$$

$$\begin{tikzcd}
\left(\boldsymbol{x}_t, \boldsymbol{y}_t\right) \arrow[r, rightsquigarrow] \arrow[d, "q(\boldsymbol{x}_t, \boldsymbol{y}_t)"'] & \left(\boldsymbol{x}_t, \boldsymbol{z}_t\right) \arrow[r, Rightarrow, "\text{FTPL}"'] \arrow[d, "\langle \boldsymbol{x}_t, \boldsymbol{z}_t \rangle"'] & \left(\hat{\boldsymbol{z}}_t\right) \arrow[r, Rightarrow, "\hat{\boldsymbol{y}}_t = \psi(\hat{\boldsymbol{z}}_t)"] \arrow[d] & \left(\hat{\boldsymbol{y}}_t\right) \arrow[d] \\
\text{Non-linear} & \text{Linear} & \text{Virtual Policy} & \text{Physical Policy}
\end{tikzcd}$$

Achievability and Converse

Theorem (Achievability)

The expected regret for the LeadCache policy is upper bounded by:

$$\mathbb{E}(R_T^{LeadCache}) \leq kn^{3/4}d^{1/4}\sqrt{mCT},$$

where $k = O(\text{poly-log}(N/C))$, n denotes the number of users and d is the maximum number of connected users per cache.

Theorem (Converse)

For a large enough library of size $N \geq \max\left(2\frac{d^2 Cm}{n}, 2mC\right)$ the regret R_T^π of any online caching policy π is lower bounded by:

$$R_T^\pi \geq \max\left(\sqrt{\frac{mnCT}{2\pi}}, d\sqrt{\frac{mCT}{2\pi}}\right) - \Theta\left(\frac{1}{\sqrt{T}}\right).$$

These two Theorems, taken together, implies that LeadCache is regret optimal up to a factor of $\tilde{O}(n^{3/8})$.

Bounding the Number of Cache Refreshes

- We now consider bounding the frequency of cache-refreshes under the following stochastic assumption.
- **Stochastic Regularity Assumption:** There exists a set of non-negative numbers $\{p_f^i\}_{i \in \mathcal{I}, f \in [N]}$ such that for any $\epsilon > 0$, we have:

$$\sum_{t=1}^{\infty} \mathbb{P}\left(\left| \frac{\boldsymbol{X}_f^i(t)}{t} - p_f^i \right| \geq \epsilon \right) < \infty, \quad \forall i \in \mathcal{I}, f \in [N]. \tag{1}$$

The above condition necessarily implies (via the First Borel-Cantelli Lemma)

$$\frac{\boldsymbol{X}_f^i(t)}{t} \to p_f^i, \quad \text{a.s.}, \forall i \in \mathcal{I}, f \in [N]. \tag{2}$$

Theorem (Finite downloads)

Under the above regularity assumption, the file fetches to the caches stop after a finite time with probability 1.

Approximation Algorithm - 1 (Pipage, deterministic)

- The ILP in LeadCache is indeed **NP-Hard** in the worst-case :(
- **Approx. Algorithm 1:** Pipage rounding [Ageev and Sviridenko 2004] for rounding $\boldsymbol{y}^*$ to an integral solution
- Consider the surrogate loss function corresponding to $L(\boldsymbol{y})$

$$\phi(\boldsymbol{y}) = \sum_{i,f} (\theta^i_f(t))^+ (1 - \prod_{j \in \partial^+(i)} (1 - y^j_f))$$

- Observe that, keeping $\boldsymbol{y}^{-j}$ fixed, $\phi(\boldsymbol{y})$ is linear in each $\boldsymbol{y}^j$
- Approximation Lemma

$$L(\boldsymbol{y}) \geq \phi(\boldsymbol{y}) \geq \left(1 - (1 - \frac{1}{\Delta})^{\Delta}\right) L(\boldsymbol{y}),$$

  where $\Delta \equiv \max_{i \in \mathcal{I}} |\partial^+(i)|$.
- The Pipage procedure rounds the solution vector iteratively such that $\{\phi(\boldsymbol{y}_t)\}_{t \geq 1}$ is non-decreasing, yielding an approximation guarantee of $1 - 1/e$.
- However, Pipage does not necessarily yield a sub-linear regret guarantee.

Approximation Algorithm - 2 (Randomized)

- Algorithm 2:
  - Relax the ILP to an LP
  - Randomly sample C files from each caches using Madow's systematic sampling with the inclusion probability vector obtained from the LP

Theorem (α-sublinear regret)

The above rounding scheme runs in linear time and yields an $1 - {}^{1}/_{e}$-regret guarantee of $\tilde{O}(n^{3/4}\sqrt{dmCT})$.

Observation: Although the randomized rounding with Madow's sampling is theoretically sound, in practice, the Pipage rounding-based scheme yields better performance.

Proof Sketch for Achievability

- Following [Cohen et al. 2015], we consider the Gaussian smoothing of the support function:

$$\phi_{\eta_t}(\boldsymbol{x}) = \mathbb{E}_{\boldsymbol{\gamma}} \max_{\boldsymbol{z} \in \mathcal{Z}} \left[\langle \boldsymbol{z}, \boldsymbol{x} + \eta_t \boldsymbol{\gamma} \rangle\right]$$

- This implies that $\nabla \phi_{\eta_t}(\boldsymbol{X}_t) = \mathbb{E}\langle \hat{\boldsymbol{z}}_t, \boldsymbol{x}_t \rangle$ and an application of Stein's lemma gives

$$\left(\nabla^2 \phi_{\eta_t}(\boldsymbol{X}_t)\right)_{\boldsymbol{p},\boldsymbol{q}} = \frac{1}{\eta_t} \mathbb{E}(\hat{\boldsymbol{z}}_{\boldsymbol{p}} \gamma_{\boldsymbol{q}}),$$

  where each of the indices $\boldsymbol{p}$ and $\boldsymbol{q}$ are (user, file) tuples.

- Using Taylor's series expansion, the regret can be bounded as

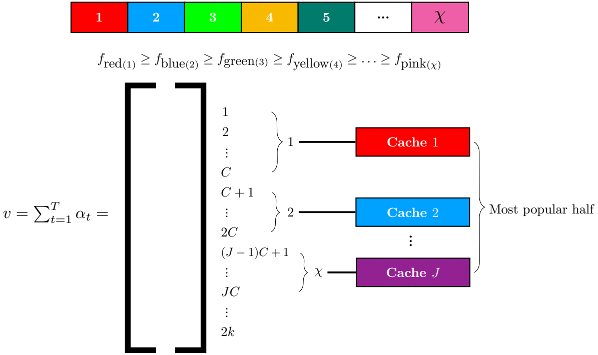
$$\mathbb{E}(R_T) \leq \underbrace{\phi_{\eta}(\boldsymbol{X}_1)}_{\text{Gaussian Complexity}} + \sum_{t=1}^{T} \left(\phi_{\eta_{t+1}}(\boldsymbol{X}_{t+1}) - \phi_{\eta_t}(\boldsymbol{X}_{t+1}) \right) + \frac{1}{2} \sum_{t=1}^{T} \boldsymbol{x}_t \nabla^2 \phi_{\eta_t}(\tilde{\boldsymbol{X}}_t) \boldsymbol{x}_t.$$

- The final regret bound is obtained by carefully bounding each of the above three terms by exploiting the structure of the problem.

Proof sketch for the Minimax Lower Bound

- We use Probabilistic methods with statistically dependent file requests - all users request the same file chosen uniformly at random
- The non-linearity of the reward function makes evaluation of the optimal offline reward OPT* challenging
  - Need to compute $\mathbb{E}(\max_{\texttt{all joint configurations}}[\texttt{ Total Hits }])$
- However, OPT* can be lower bounded by a class of cache configurations satisfying a certain local exclusivity property
  - All caches connected to each user host distinct files
- Observation: Under the local-exclusivity constraint, the reward function becomes **Linear**
- To obtain the tightest lower bound for OPT*, we need to design the best caching configuration $\boldsymbol{y}_{\perp}$ with the local exclusivity constraint

Ingredient 1: Brook's theorem for Graph Coloring

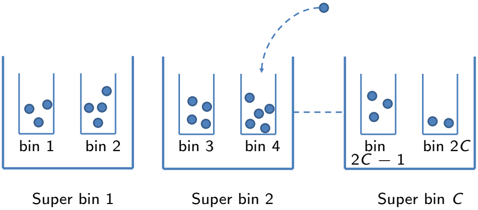


Construction of the caching configuration $\boldsymbol{y}_{\perp}$.

- Using Brook's theorem, we find a near-minimal coloring of the caches so that local exclusivity holds
- The most frequent half of the files are assigned to the caches; caches with distinct colors receive distinct files.

Ingredient 2: Balls-into-Bins

- The reward accrued by the offline optimal policy is closely related to the load $M_C(T)$ in the most-occupied C bins when T balls are thrown u.a.r. into $2C$ bins



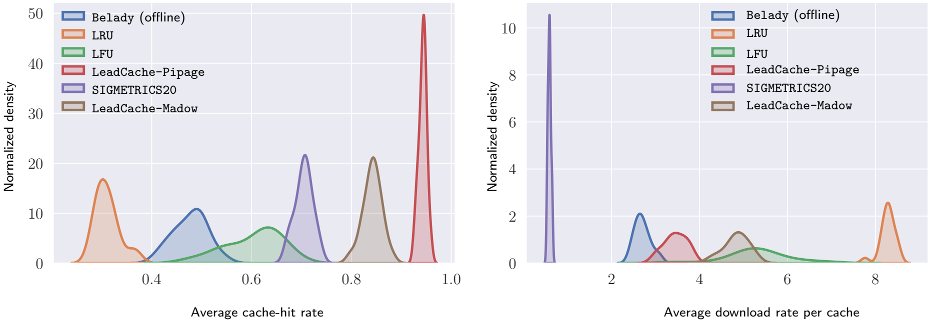
Illustrating the construction of Super bins

- To lower bound this quantity, we pair up the bins and lower bound $M_C(T)$ by the summation of max-load in each pair, which yields

$$\mathbb{E}(M_C(T)) \geq \frac{T}{2} + \sqrt{\frac{CT}{2\pi}} - \Theta(\frac{1}{\sqrt{T}}).$$

Experimental Results

Dataset: CDN trace with $\sim 375K$ requests. We consider $n = 30$ users randomly connected to $m = 15$ caches.



Impact: 1.8× increase in the hit-rates over the state-of-the-art

Code available at https://github.com/AbhishekMITIITM/LeadCache-NeurIPS21

Open Problems

- The classical notion of regret compares the performance of a policy against a clairvoyant but fixed action throughout
- In dynamic environments, fixed actions typically yield poor performance
- A stronger guarantee was obtained by Feder et al. [1992] who obtained sublinear regret guarantee against all Finite State Machine Predictors for the Binary prediction problem
    - They combine Lempel-Ziv (78) parsing with online learning methods for achieving this result
- **Problem 1:** Is it possible to extend the LeadCache policy, in particular, and OCO algorithms, in general, to have a sublinear regret guarantee against all Finite State Machines?
- **Problem 2:** If some users request unpopular content, LeadCache might virtually ignore them. How to design a network caching policy that is fair to all users?

Thanks

Thank You!

For questions, please email me at: abhishek.sinha@ee.iitm.ac.in