Regret-Optimal Online Caching for Adversarial and Stochastic Arrivals

Fathima Zarin Faizal(✉), Priya Singh, Nikhil Karamchandani, and Sharayu Moharir

Indian Institute of Technology Bombay, Mumbai 400076, Maharashtra, India
180070018@iitb.ac.in
https://iitb.ac.in

Abstract. We consider the online caching problem for a cache of limited size. In a time-slotted system, a user requests one file from a large catalog in each slot. If the requested file is cached, the policy receives a unit reward and zero rewards otherwise. We show that a Follow the Perturbed Leader (FTPL)-based anytime caching policy is simultaneously regret-optimal for both adversarial and i.i.d. stochastic arrivals. Further, in the setting where there is a cost associated with switching the cached contents, we propose a variant of FTPL that is order-optimal with respect to time for both adversarial and stochastic arrivals and has a significantly better performance compared to FTPL with respect to the switching cost for stochastic arrivals. We also show that these results can be generalized to the setting where there are constraints on the frequency with which cache contents can be changed. Finally, we validate the results obtained on various synthetic as well as real-world traces.

Keywords: Online caching · algorithms · regret bounds

1 Introduction

The caching problem has been studied since the 1960s, initially motivated by memory management in computers [21]. More recently, there has been renewed interest motivated by Content Delivery Networks [3] used for applications such as Video-on-Demand services. Such applications rely on low latency to provide a good customer experience. The framework of this problem involves a library of L files and a cache located near the end-users that is capable of storing at most C files at any given time, the algorithmic challenge being to determine the most popular files to be stored in the cache.

Two types of arrival patterns have been considered in the existing literature and in our work. The first is known as the Independent Reference Model, where

This work is supported by a SERB grant on Leveraging Edge Resources for Service Hosting.

E. Hyytiä and V. Kavitha (Eds.): VALUETOOLS 2022, LNICST 482, pp. 147–163, 2023.
https://doi.org/10.1007/978-3-031-31234-2_9

request arrivals are generated by an i.i.d. stochastic process and the distribution of the request process is unknown to the policy. The second arrival model is the adversarial arrival model where we make no structural assumptions on the arrival process. Here, the arrival pattern is generated by an oblivious adversary who knows which caching policy is being used but is not aware of the sample path of decisions made by the policy. In both models, as we are focused on the online caching problem, requests are revealed causally and therefore caching decisions have to be made based on past arrival patterns without any explicit knowledge of future arrivals.

Various metrics have been used to characterize the performance of caching polices, including hit-rate and competitive ratio. Regret is a popular metric for online learning algorithms [9] and is defined as the difference between the reward incurred by the optimal stationary policy and the policy under consideration. Our broad goal is to determine if there exists caching policies that have order-optimal regret with respect to time for i.i.d. stochastic and adversarial arrivals and therefore robust to the nature of the arrival process. Polices that perform well in the adversarial setting are primarily focused on not performing horribly on any arrival sequence. This often leads to sub-optimal performance for specific arrival sequences. Similarly, policies designed for specific arrival processes or under some structural assumptions on the arrival process have poor performance in the adversarial setting as they have very poor performance for specific arrival processes which affects the worst-case performance of the policy. For instance, policies designed for the independent reference model would not be ideal when requests are not stationary.

Prediction with expert advice [9,16] and Online Convex Optimization [23] are well-known settings in online learning for which optimal algorithms have been found. Though the caching problem is equivalent to the prediction with expert advice setting with $\binom{L}{C}$ experts, $\binom{L}{C}$ is typically a very large number resulting in standard algorithms being computationally inefficient. Least Frequently Used (LFU), Least Recently Used (LRU) and First-in-First-Out (FIFO) are popular caching policies that have been shown to achieve optimal competitive ratio [4]. There are also results on the closed form stationary hit probabilities of these algorithms under the Independent Reference Model [6,22].

Under stochastic arrivals, LFU achieves order-optimal regret [8] but under adversarial arrivals, LFU, LRU and FIFO have been shown to have suboptimal regret [20]. A sublinear regret upper bound was proved for a gradient-based coded caching policy (OGA) under adversarial arrivals [20] while the first uncoded caching policy to be shown to achieve sublinear regret is the Follow The Perturbed Leader (FTPL) policy [7]. Proposed in [11], FTPL has also been shown to achieve order-optimal regret under adversarial arrivals by proving a lower bound on the regret using a balls-into-bins argument in [7]. An FTPL-based policy has also been shown to be regret-optimal for bipartite caching networks [19].

These policies do not consider the overhead of fetching files into the cache from the library each time the cache updates, called the *switching cost* [18]. An $\tilde{O}(C\sqrt{T})$ upper bound on the regret including the switching cost was shown for a variant of the Multiplicative-Weight policy (MW) under adversarial arrivals

which is also more computationally efficient compared to the original MW algorithm naively applied to the caching problem. An upper bound of $\tilde{O}(\sqrt{CT})$ was shown for an FTPL-based policy which is also simpler to implement [18] and improves upon the earlier bound by a factor of $\Theta(\sqrt{C})$.

1.1 Our Contributions

We consider the following two settings: unrestricted switching, where the objective is to minimize the regret including the switching cost, and restricted switching, where the cache is allowed to update at certain fixed points only and the objective is to minimize regret. In Sect. 4, we consider the unrestricted switching setting and show that FTPL with an adaptive learning rate achieves order-optimal regret under stochastic arrivals even after including the switching cost, while FTPL with a constant learning rate cannot have order-optimal regret for both stochastic and adversarial arrivals. We also propose the Wait then FTPL (W-FTPL) policy that improves the bound on the switching cost from $\mathcal{O}(D)$ to $\mathcal{O}(\log D)$, where D is the per-file switching cost. In Sect. 5, we consider the restricted switching setting and prove a lower bound on the regret of any policy and an upper bound on the regret of FTPL. We show that FTPL acheives order-optimal regret under stochastic file requests and in a special case of this setting under adversarial file requests. Finally, in Sect. 6, we present the results of numerical experiments on synthetic as well as real-world traces that validate the results obtained. Due to a lack of space, the proofs of the theorems stated in this paper can be found in [1].

We thus show that FTPL with an adaptive learning rate applied to the online caching problem has order-optimal regret under stochastic and adversarial arrivals in the unrestricted switching and in a special case of the restricted switching setting.

2 Problem Formulation

We consider the classical content caching problem where a user requests files from a library that is stored in a back-end server. There is a cache that is capable of serving user requests at a lower cost but has a storage size that is typically considerably smaller than the library size. Time is slotted and in each time slot, the user requests at most one file. The sequence of events in a time slot is as follows. The cache may first update its contents, after which it receives a request for a file from the user. If the requested file is available in the cache, the cache is said to have a *hit* and the request is fulfilled locally by the cache, and otherwise a *miss*, in which case the file request is fulfilled by the back-end server.

Cache Configuration. We consider a cache of size C that stores files from a library $\mathcal{L}$ of size L. Usually, the cache size is much smaller than the library size, i.e., $C \ll L$. The file requested by the user at time t is denoted by x_t and is represented also in the form of the one-hot encoded vector $\mathbf{x}_t \in \{0,1\}^L$. For $\tau \geq 2$, we denote by $\boldsymbol{X}_\tau = \sum_{t=1}^{\tau-1} \boldsymbol{x}_t$ the L-length vector storing the cumulative sum

of requests for each file till time slot $\tau - 1$. $\boldsymbol{X}_1$ is initialized to be the zero vector. Let C(t) denote the set of files cached in round t and let $\boldsymbol{y}_t \in \{0,1\}^L$ be a binary vector denoting the state of the cache at time t, such that $y_t = (y_t^1, y_t^2, \ldots, y_t^L)$ with $y_t^i = 1$ for $i \in C(t)$ and 0 otherwise.

File Requests. We consider two types of file requests: adversarial and stochastic. The file requests are said to be *adversarial* if no assumptions are made regarding the statistical properties of the file requests. We assume that the adversary is oblivious, i.e., the entire file request sequence is fixed before the first request is sent. The file requests are said to be *stochastic* if in each slot, the request is generated independently according to a popularity distribution $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_L)$, where $\mathbb{P}(x_t = i) = \mu_i$ and $\sum_i \mu_i = 1$. Without loss of generality, we assume that $\mu_1 \geq \ldots \geq \mu_L$. As is the case in most real-world applications, the popularity distribution is assumed to be unknown to the caching policy beforehand.

Caching Policy. At the beginning of any given time slot $t \geq 2$, a caching policy $\pi(\cdot)$ maps the history of observations it has seen so far (denoted by $h(t)$) to a valid cache configuration $C(t)$, i.e., $C(t) = \pi(h(t))$. In the first time slot, we assume that the cache stores C files randomly chosen from the library and that this does not incur any switch cost. We define T to be the time horizon of interest. When the file requests are adversarial, the optimal stationary policy is defined to be the caching policy that stores the top C files in hindsight, i.e., stores the C files that received the maximum number of requests till time T. When the file requests are stochastic, the optimal stationary policy is defined to be the caching policy that stores the files with the top C popularities in the cache, i.e., $C(t) = \mathcal{C}\, \forall t$, where $\mathcal{C} = \{1, \ldots, C\}$.

Reward and Switch Cost. At each time step, the policy obtains a reward of 1 unit when the requested file is available in the cache, i.e., a hit occurs, and a reward of 0 units otherwise. A caching policy that fetches a large number of files into the cache each time the cache updates is not ideal as fetching files into the cache causes latency and consumes bandwidth. Thus, we also consider the switch cost, i.e., the cost of fetching files from the back-end server into the cache. We assume that fetching a file into the cache from the server incurs a cost of D units.

Performance Metric. Policies are evaluated on the basis of the regret that they incur. Informally, the regret of a policy till time T is the difference between the net utility of the optimal stationary policy and the net utility the policy under consideration. The net utility of a policy till time T is the difference between the net reward accrued till T and the overall switch cost incurred till then. Stationary policies do not incur any switch cost and hence their net utility is determined by the overall number of hits they have till time T. As discussed in the next section, in some cases, we omit the switch cost.

Problem Settings. We consider the following two variations of the classical content caching problem:

1. **Setting 1:** Unrestricted switching with switching cost.
   In this setting, the system incurs a cost of D units every time a file is fetched from the back-end server to be stored in the cache. When following a policy

π on a request sequence $\{x_t\}_{t=1}^T$, the regret till time T for $\{x_t\}_{t=1}^T$ including the switching cost when the file requests are adversarial is defined as:

$$R_A^\pi(\{x_t\}_{t=1}^T, T, D) = \sup_{\boldsymbol{y}\in\mathcal{Y}} \langle \boldsymbol{y}, \boldsymbol{X}_{T+1}\rangle - \sum_{t=1}^{T} \mathbb{E}\,\langle \boldsymbol{y}_t, \boldsymbol{x}_t\rangle + \frac{D}{2}\sum_{t=1}^{T-1} \mathbb{E}\left\|\boldsymbol{y}_{t+1} - \boldsymbol{y}_t\right\|_1, \tag{1}$$

where the expectation is with respect to any randomness introduced by the policy. The regret of a policy π till time T is defined as the worst-case regret over all possible request sequences, i.e.,

$$R_A^\pi(T, D) = \sup_{\{x_t\}_{t=1}^T} R_A^\pi(\{x_t\}_{t=1}^T, T, D).$$

When the file requests are stochastic, the regret including the switching cost after T time steps is defined as:

$$R_S^\pi(T, D) = \mathbb{E}\left[\sum_{t=1}^{T} \mathbb{1}\{x(t) \in \mathcal{C}\} - \mathbb{1}\{x(t) \in C(t)\}\right] + \frac{D}{2}\sum_{t=1}^{T-1} \mathbb{E}\left\|\boldsymbol{y}_{t+1} - \boldsymbol{y}_t\right\|_1, \tag{2}$$

where $\mathcal{C}$ denotes the set of files having the top C popularities. In the above expression, the expectation is taken with respect to the randomness in the file requests as well as any randomness introduced by the policy.

2. **Setting 2:** Restricted switching without switching cost.
Here, the cache is allowed to change its contents only at $s+1$ fixed time slots for some $1 \le s \le T$. To be precise, the cache is allowed to change its contents only at the beginning of the following time slots: $1, r_1+1, r_1+r_2+1, \ldots, \sum_{i=1}^{s} r_i + 1$, where $1 \le r_i \le T, 1 \le i \le s$ denotes the i^{th} inter-switching period such that $\sum_{i=1}^{s} r_i = T$. Thus, within the time horizon T, the cache is allowed to update only at s fixed time slots. Note that the setting where the cache is allowed to change its contents only after every $1 \le r \le T$ requests, i.e., at time slots $1, r+1, \ldots, T+1$ and $s = \frac{T}{r}$ is a special case of this setting. For simplicity, we restrict our attention to the case where there is no switch cost, i.e., $D = 0$. When following a policy π, the regret after T time steps when the file requests are adversarial is:

$$R_A^\pi(T) = \sup_{\boldsymbol{y}\in\mathcal{Y}} \langle \boldsymbol{y}, \boldsymbol{X}_{T+1}\rangle - \sum_{t=1}^{T} \mathbb{E}\,\langle \boldsymbol{y}_t, \boldsymbol{x}_t\rangle, \tag{3}$$

where the expectation is with respect to the randomness introduced by the policy. When the file requests are stochastic, the regret after T time steps is:

$$R_S^\pi(T) = \mathbb{E}\left[\sum_{t=1}^{T} \mathbb{1}\{x(t) \in \mathcal{C}\} - \mathbb{1}\{x(t) \in C(t)\}\right], \tag{4}$$

where $\mathcal{C}$ denotes the set of files having the top C popularities. In the above expression, the expectation is taken with respect to the randomness introduced by the policy and the file requests.

To distinguish between results including switching cost and those without switching cost, we use the notation $R^{\pi}_{(\cdot)}(T, D)$ for results involving the switch cost and $R^{\pi}_{(\cdot)}(T)$ for results without the switch cost.

The overall goal of this work is to characterize the optimal regret in the two settings mentioned above, for both adversarial and stochastic file requests. This entails proving scheme-agnostic lower bounds on the regret as well as designing policies whose regret is of the same order as these lower bounds. As we will see, these results will also highlight the impact of switching cost and intermittent switching on the optimal achievable regret.

3 Policies

In this section, we introduce and formalize policies whose optimality (or suboptimality) will be discussed in later sections.

3.1 Least Frequently Used (LFU)

The LFU algorithm (formally defined in Algorithm 1) keeps track of the number of times each file has been requested so far. At each time step t, the files with the C highest number of requests are cached. This policy is deterministic and thus performs poorly when faced with certain adversarial request sequences [20]. For the simplified case of $L = 2, C = 1$, one example is a round-robin request sequence of the form $1, 2, 1, 2, \ldots$ which would result in LFU obtaining essentially zero reward while the optimal stationary policy obtains a reward of $T/2$. For stochastic requests, it has been shown to achieve $\mathcal{O}(1)$ regret when switching is allowed at all time slots and when the algorithm incurs no switch cost [8].

Algorithm 1. LFU algorithm

```
procedure LFU(T)
    c_t ← 0
    while t ≤ T do
        C_t ← arg max_C (c_t(1), ..., c_t(L))
        Receive file request x_t
        c_t(x_t) ← c_t(x_t) + 1
    end while
end procedure
```

3.2 Follow the Perturbed Leader (FTPL)

The FTPL algorithm (formally defined in Algorithm 2) is a variation of the LFU algorithm and also keeps track of the number of times each file has been

requested so far, but adds an independent Gaussian perturbation with mean 0 and standard deviation η_t (referred to as the learning rate) to the counts of each file in each time slot. At each time step t, the files with the C highest perturbed counts are cached. Special cases of this policy are known to achieve order-optimal regret under adversarial requests (with and without switch cost) [7,18]. We will henceforth refer to the FTPL algorithm with the learning rate η_t by FTPL(η_t).

Algorithm 2. FTPL algorithm

procedure FTPL($T, \{\eta_t\}_{t=1}^T$)
  $\boldsymbol{c}_t \leftarrow \mathbf{0}$
  Sample $\boldsymbol{\gamma} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}_{L\times L})$
  while $t \leq T$ **do**
    $C_t \leftarrow \arg\max_C (c_t(1) + \eta_t\gamma(1), \ldots, c_t(L) + \eta_t\gamma(L))$
    Receive file request x_t
    $c_t(x_t) \leftarrow c_t(x_t) + 1$
  end while
end procedure

3.3 Wait Then FTPL (W-FTPL)

The algorithm that we propose, Wait then FTPL (formally defined in Algorithm 3), is a variant of the FTPL algorithm where the policy remains idle for an initial deterministic waiting period and then follows the normal FTPL algorithm. The motivation for this algorithm is to avoid the higher switch cost incurred initially by the FTPL algorithm under stochastic file requests until the policy has seen enough requests to have a good enough estimate of the underlying popularity distribution, while ensuring order-optimal regret in the adversarial setting. We will henceforth refer to the W-FTPL algorithm with the learning rate η_t by W-FTPL(η_t).

Algorithm 3. W-FTPL algorithm

procedure W-FTPL($T, \{\eta_t\}_{t=1}^T, D, t'$)
  $\boldsymbol{c}_t \leftarrow \mathbf{0}$
  Sample $\boldsymbol{\gamma} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}_{L\times L})$
  while $t \leq T$ **do**
    if $t > t'$ **then**
      $C_t \leftarrow \arg\max_C (c_t(1) + \eta_t\gamma(1), \ldots, c_t(L) + \eta_t\gamma(L))$
    end if
    Receive file request x_t
    $c_t(x_t) \leftarrow c_t(x_t) + 1$
  end while
end procedure

4 Setting 1: Unrestricted Switching with Switching Cost

In this section, we consider the setting where there is no limitation on the switching frequency of the cache and the objective is to minimize the regret including the switching cost, i.e., minimize regret as well as the number of fetches into the cache. We consider both stochastic and adversarial file request sequences and show that FTPL($\alpha\sqrt{t}$) and W-FTPL($\alpha\sqrt{t}$) are order-optimal under both types of file requests. While the FTPL(η) algorithm is order-optimal under adversarial requests for a particular value of η [7], we prove that the same does not hold true for stochastic file requests.

4.1 Adversarial Requests

In this section, we discuss the performance of the policies introduced in Sect. 3 under adversarial file requests. The key results of this section has been summarized in the following theorem:

Theorem 1. *Under adversarial requests, we have*

(a) [7, Theorem 2] For any policy π and $L \geq 2C$,

$$R_A^{\pi}(T, D = 0) \geq \sqrt{\frac{CT}{2\pi}} - \Theta\left(\frac{1}{\sqrt{T}}\right).$$

(b) [20, Proposition 1] The regret of the LFU policy can be characterized as:

$$R_A^{LFU}(T, 0) = \Omega(T).$$

(c) [18, Theorem 4.1] The regret of FTPL($\alpha\sqrt{t}$) is upper bounded as:

$$R_A^{FTPL(\alpha\sqrt{t})}(T, D) \leq c_1\sqrt{T} + c_2 \ln T + c_3,$$

where $c_1 = \mathcal{O}(\sqrt{\ln(Le/C)})$, and c_2, c_3 are small constants depending on L, C, D and α.

(d) The regret of W-FTPL($\alpha\sqrt{t}$) is upper bounded as:

$$R_A^{W\text{-}FTPL(\alpha\sqrt{t})}(T, D) \leq \mathcal{O}(\sqrt{T}).$$

Part (a) has been proved in [7] and provides a lower bound on the regret of any policy under adversarial requests.

As argued before, LFU performs poorly under adversarial requests. This is also seen for many popular classical caching algorithms like LRU and FIFO (refer [20]).

Part (c) has been proved in [18] and provides an $\mathcal{O}(\sqrt{T})$ upper bound on the regret including the switching cost of the FTPL($\alpha\sqrt{t}$) policy under adversarial requests, thus showing that this algorithm is order-optimal under adversarial requests. FTPL($\alpha\sqrt{T}$) has also been shown to be order-optimal under adversarial requests [7,18].

Part (d) provides an upper bound on the regret including the switching cost of W-FTPL($\alpha\sqrt{t}$) under adversarial requests. This result shows that W-FTPL($\alpha\sqrt{t}$) is order-optimal under adversarial requests. The proof of this result can be found in Appendix A.

4.2 Stochastic Requests

To find a policy that achieves order-optimal regret under stochastic and adversarial arrivals, we were motivated by [7,18] where the regret for FTPL with the learning rates $\alpha\sqrt{t}$ and $\alpha\sqrt{T}$ (α being some positive constant) under adversarial arrivals was characterized. In this section, we discuss the performance of these policies under stochastic file requests. The key results of this section has been summarized in the following theorem:

Theorem 2. *The file requests are stochastic i.i.d. with the popularity distribution* $\boldsymbol{\mu}$.

(a) [8, Theorem 1] When $D = 0$, *the regret of the LFU policy can be upper bounded as:*

$$R_S^{LFU}(T,0) < \min\left(\frac{16}{\Delta_{\min}^2}, \frac{4\,C(L-C)}{\Delta_{\min}}\right), \tag{5}$$

where $\Delta_{\min} = \mu_C - \mu_{C+1}$.

(b) For $L = 2, C = 1$ *and* $D = 0$, *the regret of FTPL(*η*) can be lower bounded as:*

$$R_S^{FTPL(\eta)}(T,0) \geq \frac{\eta e^{-\left(\frac{1+\eta}{\eta}\right)^2}}{4}.$$

(c) The regret of FTPL($\alpha\sqrt{t}$*) is upper bounded as:*

$$R_S^{FTPL(\sqrt{t})}(T,D) \leq (1+DC)\,t_0 + \left(1+\frac{D}{\Delta_{\min}}\right)\left(\frac{8}{\Delta_{\min}} + \frac{32\alpha^2}{\Delta_{\min}}\right),$$

where $t_0 = \max\left\{\frac{8}{\Delta_{\min}^2}\log\left(L^3\right), \frac{32\alpha^2}{\Delta_{\min}^2}\log\left(L^3\right)\right\}$.

(d) The regret of W-FTPL($\alpha\sqrt{t}$*) is upper bounded as:*

$$R_S^{W\text{-}FTPL(\sqrt{t})}(T,D) \leq t' + \frac{16}{\Delta_{\min}} + \frac{64\alpha^2}{\Delta_{\min}} + 2L^3D\left(e^{-u(\log D)^{1+\beta}\Delta_{\min}^2/8}\frac{8}{\Delta_{\min}^2} + e^{-u(\log D)^{1+\beta}\Delta_{\min}^2/32\alpha^2}\frac{32\alpha^2}{\Delta_{\min}^2}\right),$$

where $t' = \max\left\{\frac{8}{\Delta_{\min}^2}\log\left(\frac{L^3}{2}\right), \frac{32\alpha^2}{\Delta_{\min}^2}\log\left(\frac{L^3}{2}\right), u(\log D)^{1+\beta}\right\}$.

Part (a) of the above theorem has been proved in [8] and shows that the regret of LFU is $\mathcal{O}(1)$ when the file requests are stochastic. Thus, the regret of any policy that has order optimal regret under stochastic file requests should be $\mathcal{O}(1)$.

Part (b) gives a lower bound on the regret of the FTPL(η) algorithm under stochastic file requests. Note that for all $\eta \geq 1$, we have

$$R_S^{\text{FTPL}(\eta)}(T) \geq \frac{\eta e^{-\left(\frac{1+\eta}{\eta}\right)^2}}{4} \geq \frac{\eta}{4e^4}.$$

This shows that the regret is $\Omega(\eta)$ for the FTPL(η) algorithm when the file requests are stochastic. Using $\eta = \alpha\sqrt{T}$, where α is a positive constant, from [7] which gave an $\mathcal{O}(\sqrt{T})$ upper bound for FTPL(η), we get that with the constant learning rate $\eta = \alpha\sqrt{T}$, the regret of FTPL(η) is $\Theta(\sqrt{T})$. Thus, while FTPL(η) is order-optimal under adversarial requests (see Sect. 4.1), it cannot simultaneously be order-optimal under adversarial and stochastic file requests. The proof of this result can be found in Appendix B.

Part (c) shows that FTPL($\alpha\sqrt{t}$), $\alpha > 0$ is order-optimal when the file requests are stochastic. While FTPL(η) with $\eta = \mathcal{O}(\sqrt{T})$ achieves $\Omega(\sqrt{T})$ regret, this result shows an $\mathcal{O}(1)$ upper bound on the regret of FTPL($\alpha\sqrt{t}$) including the switching cost, thus showing that this algorithm is order-optimal. Note that the upper bound grows linearly with the per-file switch cost D. The proof of this result can be found in Appendix C.

Recall that the Wait then FTPL($\alpha\sqrt{t}$) (W-FTPL($\alpha\sqrt{t}$)) algorithm is a variant of the FTPL($\alpha\sqrt{t}$) algorithm. The algorithm remains idle till time $t' = u(\log D)^{1+\beta}, u > 0$, and then normal FTPL($\alpha\sqrt{t}$) is followed. Part (d) proves an $\mathcal{O}(1)$ upper bound on the regret including the switching cost of this algorithm with respect to the horizon T under stochastic file requests, thus showing that this algorithm is order-optimal. The main improvement over the FTPL($\alpha\sqrt{t}$) algorithm is the $\mathcal{O}((\log D)^{1+\beta})$ upper bound on the regret including the switching cost for a large enough value of D under stochastic file requests, as compared to the upper bound $\mathcal{O}(D)$ for FTPL($\alpha\sqrt{t}$). The key idea behind remaining idle for an initial period that depends logarithmically on D is to avoid the higher switch cost incurred at the beginning by the FTPL($\alpha\sqrt{t}$) algorithm (refer to Sect. 6). The proof of this result can be found in Appendix D.

5 Restricted Switching

In this section, we consider the setting where the cache is allowed to update its contents only at $s+1$ fixed number of time slots, where $s \in \mathbb{Z}, s \leq T$. The first point is at time slot 1, the second at time slot $r_1 + 1$, the third point is at time slot $r_1 + r_2 + 1$, and so on till the $s + 1^{\text{th}}$ point, which is at time slot $\sum_{i=1}^{s} r_i + 1$, where $r_i \in \mathbb{Z}, r_i \leq T, 1 \leq i \leq s$ and $\sum_{i=1}^{s} r_i = T$. Note that the cache is allowed to update its contents only s times till the time horizon T. Refer to Fig. 1 for an illustration of this setting. As a special case of this setting, we also consider the homogenous case where the cache is allowed to update only after every $r \in \mathbb{Z}$ requests, i.e., $s = \frac{T}{r}$. We study the regret performance of FTPL and also provide lower bounds on the regret incurred by any online scheme. In the homogenous case, we also show that FTPL($\sqrt{rt}$) achieves order-optimal regret.

5.1 Stochastic Requests

Theorem 3. *The file requests are stochastic i.i.d. with the popularity distribution* $\boldsymbol{\mu}$*. When cache updates are restricted to* $s+1$ *fixed points defined by the inter-switching periods* $\{r_i\}_{i=1}^{s}$ *as outlined above,*

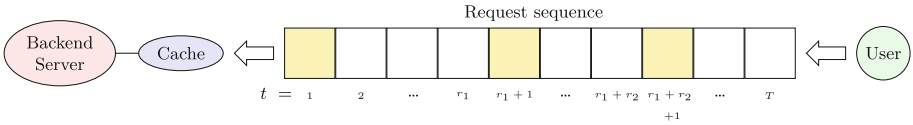

Fig. 1. The time slots where the cache is allowed to update its contents have been marked in yellow. (Color figure online)

(a) When $L = 2, C = 1$, for any online caching policy π, there exists a popularity distribution such that the popularities of the two files are greater than $1 > a > 0$ and the difference in the popularities is Δ, such that

$$R_S^{\pi}(T) \geq \frac{r_1 \Delta}{2} + \sum_{i=2}^{s} r_i \frac{\Delta}{4} \exp\left(-t_i \frac{\Delta^2}{a^2}\right).$$

(b) The regret of FTPL($\alpha\sqrt{t}$) is upper bounded as:

$$R_S^{FTPL(\alpha\sqrt{t})}(T) \leq r_1 + 2 \sum_{j=1}^{C} \sum_{k=C+1}^{L} \sum_{i=2}^{s} r_i \, \Delta_{j,k} \left(e^{-t_i \Delta_{j,k}^2/8} + e^{-t_i \Delta_{j,k}^2/32\alpha^2}\right).$$

In part (a), we prove a fundamental lower bound on the regret of any policy π under stochastic file requests when cache updates are restricted to $s + 1$ fixed time slots. The proof of this result can be found in Appendix E. In part (b), we prove an upper bound on the regret of the FTPL($\alpha\sqrt{t}$) policy when cache updates are restricted to $s + 1$ fixed time slots. The proof of this result can be found in Appendix F. We thus have that the FTPL($\alpha\sqrt{t}$) policy has order-optimal regret in this setting under stochastic file requests. Next, we consider the special case where all r_i are equal to r, i.e., $s = T/r$.

Theorem 4. *The file requests are stochastic i.i.d. with the popularity distribution $\boldsymbol{\mu}$. When the cache is allowed to update only after every r requests,*

$$R_S^{FTPL(\alpha\sqrt{t})} \leq 1 + t_0' + 2\left(\frac{8}{\Delta_{\min}} + \frac{32\alpha^2}{\Delta_{\min}}\right).$$

where $t_0' = \max\left\{r, \frac{8}{\Delta_{\min}^2} \log\left(L^2\right), \frac{32\alpha^2}{\Delta_{\min}^2} \log\left(L^2\right)\right\}$.

In Theorem 4, we prove an $\mathcal{O}(\max\{r, \log L\})$ upper bound on the regret of the FTPL($\alpha\sqrt{t}$) algorithm under stochastic file requests. While the order-optimality of this policy with respect to r follows from Theorem 3, we also note that the bound proved here improves upon the worst-case $\mathcal{O}(L^2)$ dependency in the upper bound proved in part (b) of Theorem 3 to $\mathcal{O}(\log L)$. The proof of this result can be found in Appendix G.

5.2 Adversarial Requests

Theorem 5. *Files are requested by an oblivious adversary. When cache updates are restricted to $s+1$ fixed points defined by $\{r_i\}_{i=1}^{s}$ as outlined above,*

(a) For any online caching policy π and $L \geq 2C$,

$$R_A^{\pi}(T) \geq \frac{1}{2}\left(0.15\sqrt{C\sum_{i=1}^{s} r_i^2}\left(1-\frac{(C-1)\left(\sum_{i=1}^{s} r_i^4\right)}{2\left(\sum_{i=1}^{s} r_i^2\right)^2}\right) - 0.6\,C\max_{1\leq i\leq s} r_i\right).$$

(b) The regret of FTPL($\alpha\sqrt{t}$) is upper bounded as:

$$R_A^{FTPL(\alpha\sqrt{t})}(T) \leq \mathcal{O}(\alpha\sqrt{T}) + \sqrt{\frac{2}{\pi}}\sum_{i=1}^{s}\frac{r_i^2}{\alpha\sqrt{\sum_{j=0}^{i-1} r_j}}.$$

The proof of this theorem can be found in Appendix H and Appendix I respectively. In part (a), we prove a lower bound on the regret of any policy π under adversarial file requests when cache updates are restricted to s fixed time slots. Note that a necessary condition for this bound to be meaningful is

$$4\max_{1\leq i\leq s} r_i \leq \sqrt{\sum_{i=1}^{s} r_i^2}. \tag{6}$$

When this bound is meaningful, we have that $R_A^{\pi}(T) = \Omega\left(\sqrt{C\sum_{i=1}^{s} r_i^2}\right)$ for any online caching policy π. When all the r_i's are equal, this condition translates to $r \leq T/16$. This condition does not hold if any of the r_i's is too large. For instance, when $s=3$ and $r_1 = T/2, r_2 = r_3 = T/4$, this condition does not hold. When $\max\limits_{1\leq i\leq s} r_i$ and $\min\limits_{1\leq i\leq s} r_i$ are known, a sufficient condition for (6) to hold is:

$$\frac{\max\limits_{1\leq i\leq s} r_i}{\min\limits_{1\leq i\leq s} r_i} \leq \frac{\sqrt{s}}{4}.$$

We now discuss the special case where all r_i are equal to r, i.e., $s = T/r$. It follows from part (b) of Theorem 5 that FTPL($\sqrt{rt}$) achieves a regret of $\mathcal{O}(\sqrt{rT})$. The following theorem provides a matching lower bound for this setting that proves that FTPL($\sqrt{rt}$) is order-optimal, the proof of which can be found in Appendix J.

Theorem 6. *Files are requested by an oblivious adversary. When the cache is allowed to update only after every r requests, for any online caching policy π and $L \geq 2C$,*

$$R_A^{\pi}(T) \geq \begin{cases} \sqrt{\frac{CrT}{2\pi}} - \Theta\left(\frac{r\sqrt{r}}{\sqrt{T}}\right), & \text{when } r = o(T), \\ \Omega(T), & \text{when } r = \Omega(T). \end{cases}$$

6 Numerical Experiments

In this section, we present the results of numerical simulations for the various policies discussed in Sect. 3.

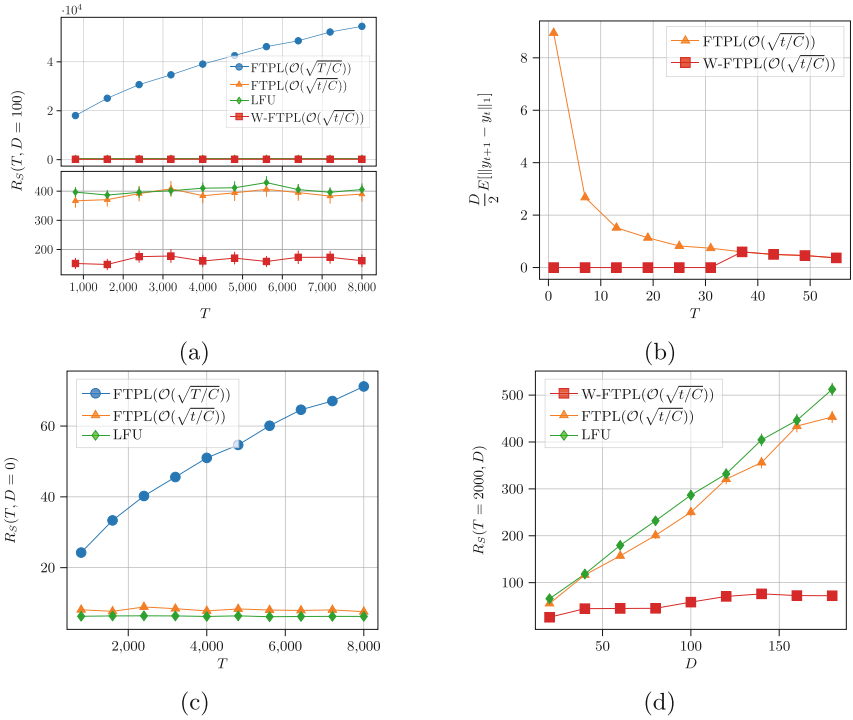

Fig. 2. The plots compare (a) the regret including the switching cost for $D = 100$ as a function of T, (b) the switching cost in each time slot for $D = 30, u = 5, \beta = 0.6$ as a function of T, (c) the regret as a function of T with $D = 0$, and (d) the regret including the switching cost for $T = 2000$ as a function of D, of various caching policies under stochastic file requests. $L = 10, C = 4$ for each of the plots and the popularity distribution is a dyadic distribution. Parts (a) and (c) show that the regret incurred by FTPL($\mathcal{O}(\sqrt{T})$) is increasing with T while FTPL($\mathcal{O}(\sqrt{t})$), W- FTPL($\mathcal{O}(\sqrt{t})$) and LFU have essentially constant regret. Part (b) shows that FTPL($\mathcal{O}(\sqrt{t})$) makes more switches at the beginning, thus motivating the W- FTPL($\mathcal{O}(\sqrt{t})$) algorithm. Part (d) shows that while the regret including the switching cost of LFU and FTPL($\mathcal{O}(\sqrt{t})$) increase linearly in D, it increases sublinearly for W-FTPL($\mathcal{O}(\sqrt{t})$).

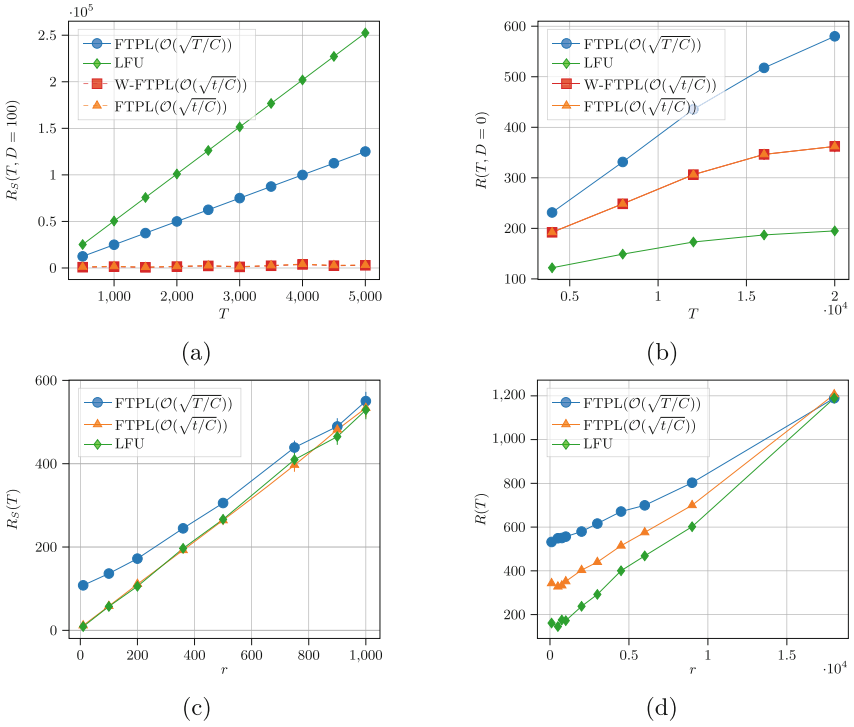

Fig. 3. The plots compare (a) the regret including the switching cost for $D = 100$ as a function of T on a round robin request sequence, (b) the regret without the switching cost, i.e., $D = 0$ as a function of T on the MovieLens dataset, (c) the regret as a function of the constant switching frequency r under stochastic file requests from a dyadic distribution, and (d) the regret as a function of the switching frequency r on the MovieLens dataset, of various caching policies. We used $L = 2, C = 1$ for Part (a) and $L = 10, C = 4$ for Part (c). In Part (a), note that the regret including the switching cost scales linearly with T for LFU, while W-FTPL($\mathcal{O}(\sqrt{t})$) and FTPL($\mathcal{O}(\sqrt{t})$) show better performance. In Part (c), the regret scales linearly with r for all the three algorithms as expected. The regret scales sublinearly with T in Part (b) and linearly with r in Part (d).

6.1 Setting 1 with Stochastic File Requests

Setup. We use $L = 10, C = 4$ throughout this section. The popularity distribution used is a dyadic distribution, i.e., for $1 \leq i \leq L-1$, $\mu(i) = \frac{1}{2^i}$ and $\mu(L) = \frac{1}{2^{L-1}}$.

Results. Fig. 2a shows that the regret of FTPL including the switching cost increases with T, while it is essentially constant for FTPL($\mathcal{O}(\sqrt{t})$), W-FTPL($\mathcal{O}(\sqrt{t})$) and LFU. One can also observe that W- FTPL($\mathcal{O}(\sqrt{t})$) performs the best among all the four algorithms. In Fig. 2c, we plot only the regret as a function of T. Note that the same trend is observed here as well. Here, we

omit plotting W- FTPL($\mathcal{O}(\sqrt{t})$) as its regret would be the same as that of FTPL($\mathcal{O}(\sqrt{t})$) in this case. Figure 2b shows that FTPL($\mathcal{O}(\sqrt{t})$) indeed makes more switches at the beginning, which is the motivation for the W-FTPL($\mathcal{O}(\sqrt{t})$) policy where no switches are made for an initial period. Figure 2d shows that the regret of LFU and FTPL($\mathcal{O}(\sqrt{t})$) grows linearly with D, while that of W-FTPL($\mathcal{O}(\sqrt{t})$) grows sublinearly with D.

6.2 Setting 1 with Adversarial File Requests

Setup. We consider a synthetic adversarial request sequence in Fig. 3a and a real-world trace in Fig. 3b. The synthetic adversarial request sequence used is $1, 2, 1, 2, \ldots$ for $L = 2, C = 1$, i.e., a round-robin request sequence. The real-world trace used is the first 20,000 rows of the MovieLens 1M dataset [2,12] which contains ratings for 2569 movies with timestamps that we model as requests to a CDN server of library size 2569 and a cache size of 25.

Results. Fig. 3a shows that under the round-robin request sequence, the regret including the switching cost of LFU scales linearly with T, while that of W-FTPL($\mathcal{O}(\sqrt{t})$) and FTPL($\mathcal{O}(\sqrt{t})$) scales sublinearly with T. Figure 3b shows that on the MovieLens dataset, the regret scales sublinearly with T for all the four algorithms.

6.3 Setting 2

Setup. We consider stochastic file requests drawn from a dyadic distribution for $L = 10, C = 4$ in Fig. 3c and file requests from the MovieLens dataset in Fig. 3d. We also used $T = 18000$ and chose r to be factors of T. There were 2518 unique movies in the first 18000 rows of the MovieLens dataset and we set the cache size to be 25.

Results. Figure 3c shows that when file requests are drawn from a dyadic popularity distribution, the regret of all three policies vary linearly with r. Figure 3d shows that on the MovieLens dataset, the regret scales linearly with r for all the three policies.

7 Conclusion

We have shown that FTPL($\mathcal{O}(\sqrt{t})$) achieves order-optimal regret even after including the switching cost under stochastic requests. Combining with prior results on the performance of FTPL, it is simultaneously order-optimal for both stochastic and adversarial requests. We also showed that FTPL(η) cannot possibly achieve order-optimal regret simultaneously under both stochastic and adversarial requests, while variants of this policy can individually be order-optimal under each type of request. We proposed the W-FTPL($\mathcal{O}(\sqrt{t})$) policy as a way of preventing the high switching cost incurred by FTPL at the beginning under stochastic file requests. We also considered the restricted switching setting, where

the cache is allowed to update its contents only at specific pre-determined time slots and obtained a lower bound on the regret incurred by any policy. We proved an upper bound on the regret of FTPL($\mathcal{O}(\sqrt{t})$) and showed that it is order-optimal under stochastic file requests and in the homogenous restricted switching case under adversarial file requests.

This work motivates several directions for future work: (1) Bringing the upper and lower bounds closer in the general restricted switching setting would help in proving whether FTPL($\mathcal{O}(\sqrt{t})$) is order-optimal or not under adversarial file requests in this case too. (2) For the restricted switching setting, our results consider only the regret. Adding the switching cost too here would make the bounds complete.

References

1. https://github.com/fathimazarin/Valuetools2022_caching/blob/main/main.pdf
2. MovieLens 1M dataset. https://grouplens.org/datasets/movielens/. Accessed 8 Aug 2022
3. Aggarwal, C., Wolf, J.L., Yu, P.S.: Caching on the world wide web. 125 J. Distrib. Parallel Syst. (IJDPS) **2**(6), 94–107 (2000). https://doi.org/10.1109/69.755618
4. Albers, S.: Competitive Online Algorithms. BRICS, Shanghai (1996)
5. Alon, N., Spencer, J.: Appendix B: Paul Erdös. John Wiley and Sons, Ltd, New York (2008). https://doi.org/10.1002/9780470277331.app2
6. Aven, O.I., Coffman, E.G., Kogan, Y.A.: Stochastic Analysis of Computer Storage. Kluwer Academic Publishers, USA (1987)
7. Bhattacharjee, R., Banerjee, S., Sinha, A.: Fundamental limits of online network-caching. CoRR abs/2003.14085 (2020). https://doi.org/10.48550/arXiv.2003.14085
8. Bura, A., Rengarajan, D., Kalathil, D., Shakkottai, S., Chamberland, J.F.: Learning to cache and caching to learn: regret analysis of caching algorithms. IEEE/ACM Trans. Netw. **30**(1), 18–31 (2022). https://doi.org/10.48550/arXiv.2004.00472
9. Cesa-Bianchi, N., Lugosi, G.: Prediction, Learning, and Games. Cambridge University Press, Cambridge (2006). https://doi.org/10.1017/CBO9780511546921
10. Cohen, A., Hazan, T.: Following the perturbed leader for online structured learning. In: Bach, F., Blei, D. (eds.) Proceedings of the 32nd International Conference on Machine Learning. Proceedings of Machine Learning Research, PMLR, Lille, France, vol. 37, pp. 1034–1042, 07–09 July 2015
11. Hannan, J.: Approximation to Bayes risk in repeated play. In: Contributions to the Theory of Games, vol. 3, no. 2, 97–139 (1957). https://doi.org/10.1515/9781400882151
12. Harper, F.M., Konstan, J.A.: The Movielens datasets: history and context. ACM Trans. Interact. Intell. Syst. **5**(4), December 2015. https://doi.org/10.1145/2827872
13. Hoeffding, W.: Probability inequalities for sums of bounded random variables. J. Am. Stat. Assoc. **58**(301), 13–30 (1963). https://doi.org/10.1080/01621459.1963.10500830
14. Kaas, R., Buhrman, J.M.: Mean, median and mode in binomial distributions. Stat. Neerl. **34**(1), 13–18 (1980). https://doi.org/10.1111/j.1467-9574.1980.tb00681.x
15. Lattimore, T., Szepesvári, C.: Bandit Algorithms. Cambridge University Press, Cambridge (2020). https://doi.org/10.1017/9781108571401

16. Littlestone, N., Warmuth, M.K.: The weighted majority algorithm. Inf. Comput. **108**(2), 212–261 (1994). https://doi.org/10.1006/inco.1994.1009
17. Mourtada, J., Gaïffas, S.: On the optimality of the hedge algorithm in the stochastic regime. J. Mach. Learn. Res. **20**, 1–28 (2019). https://doi.org/10.48550/arXiv.1809.01382
18. Mukhopadhyay, S., Sinha, A.: Online caching with optimal switching regret. CoRR abs/2101.07043 (2021). https://doi.org/10.1109/ISIT45174.2021.9517925
19. Paria, D., Sinha, A.: Leadcache: regret-optimal caching in networks. In: Thirty-Fifth Conference on Neural Information Processing Systems, vol. 34, pp. 4435–4447 (2021). https://doi.org/10.48550/arXiv.2009.08228
20. Paschos, G.S., Destounis, A., Vigneri, L., Iosifidis, G.: Learning to cache with no regrets. In: IEEE INFOCOM 2019 - IEEE Conference on Computer Communications, pp. 235–243. IEEE Press (2019). https://doi.org/10.1109/INFOCOM.2019.8737446
21. Silberschatz, A., Galvin, P., Gagne, G.: Operating System Principles. John Wiley & Sons, Hoboken (2006)
22. Starobinski, D., Tse, D.: Probabilistic methods for web caching. Perform. Eval. **46**(2–3), 125–137, October 2001. https://doi.org/10.1016/S0166-5316(01)00045-1
23. Zinkevich, M.: Online convex programming and generalized infinitesimal gradient ascent. In: Proceedings of the 20th International Conference on Machine Learning (ICML-03), pp. 928–936 (2003)