Optimistic No-regret Algorithms for Discrete Caching

NARAM MHAISEN, Delft University of Technology, The Netherlands
ABHISHEK SINHA, Tata Institute of Fundamental Research, India
GEORGIOS PASCHOS, Amazon, Luxembourg
GEORGE IOSIFIDIS, Amazon & Delft University of Technology, The Netherlands

We take a systematic look at the problem of storing whole files in a cache with limited capacity in the context of optimistic learning, where the caching policy has access to a prediction oracle (provided by, e.g., a Neural Network). The successive file requests are assumed to be generated by an adversary, and no assumption is made on the accuracy of the oracle. In this setting, we provide a universal lower bound for prediction-assisted online caching and proceed to design a suite of policies with a range of performance-complexity trade-offs. All proposed policies offer sublinear regret bounds commensurate with the accuracy of the oracle. Our results substantially improve upon all recently-proposed online caching policies, which, being unable to exploit the oracle predictions, offer only $O(\sqrt{T})$ regret. In this pursuit, we design, to the best of our knowledge, the first comprehensive optimistic Follow-the-Perturbed leader policy, which generalizes beyond the caching problem. We also study the problem of caching files with different sizes and the bipartite network caching problem. Finally, we evaluate the efficacy of the proposed policies through extensive numerical experiments using real-world traces.

CCS Concepts: • **Networks → Network performance analysis**.

1 INTRODUCTION

This paper addresses the discrete caching (prefetching) problem: choose files to replicate in a local cache in order to maximize the probability that a new file request is served locally. *Hitting* the cache speeds up CPU, optimizes user experience in CDN's [1], and enhances the performance of wireless networks [2]. With the perpetual growth of Internet traffic fueled by new services such as AR/VR [3], caching policies that learn fast to maximize cache hits can mitigate the increasing costs of information transportation [4], and similar benefits can be expected for embedded and other computing systems [5]. This works aspires to advance our theoretical understanding of this fundamental problem and proposes new provably-optimal and computationally-efficient caching algorithms using a new modeling and solution approach based on *optimistic learning*.

1.1 Motivation

Common caching policies store the newly requested files and employ the Least-Recently-Used (LRU) [6], Least-Frequently-Used (LFU) [7] and other similar rules to evict files when the cache capacity is exhausted. Under certain statistical assumptions on the request trace, such policies maintain the cache at an optimal state, see [8, Sec. 3.1-3.2]. However, with frequent addition of new content to libraries of online services and the high volatility of file popularity [9], these policies can perform arbitrarily bad. This has spurred intensive research efforts for policies that operate under more general conditions by learning on-the-fly the request distribution or adapting dynamically, e.g., with Reinforcement Learning, to observed requests; see Sec. 2. Nonetheless, these studies do not offer performance guarantees nor scale well for large libraries. The goal of this work is to design robust caching policies that are able to learn effective caching decisions with the aid of a prediction oracle of *unknown quality* (Fig. 1 left) even when the file requests are made in an adversarial fashion.

To that end, we formulate the caching problem as an online convex optimization (OCO) problem [10]. At each slot $t = 1, 2, \ldots, T$, a learner (the caching policy) selects a caching vector $x_t \in \mathcal{X}$ from the set of admissible cache states $\mathcal{X} \subseteq \{0, 1\}^N$ for a cache of size C, where N is the library size. Then, a 1-hot vector $\theta_t \in \{0, 1\}^N$ with value 1 for the requested file is revealed, and the learner
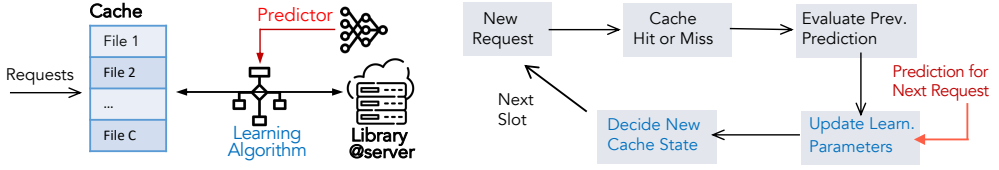
Fig. 1. Optimistic online caching with predictions: system schematic (*left*) & algorithm template (*right*).

receives a reward of $f_t(x_t) = \langle \theta_t, x_t \rangle$ for cache hits. The reward is revealed only after committing x_t, which naturally matches the dynamic caching operation where the cached files are decided before the next request arrives. Here, the learner makes no statistical assumptions and θ_t can follow any distribution, even one that is handpicked by an adversary. In the optimistic framework, the learner does not only consider its hit or miss performance so far when deciding x_t, but also the predictor's performance and output (Fig. 1 right). As customary in the online learning literature, we characterize the policy's performance by using the static *regret* metric:

$$R_T(\{x\}_T) \triangleq \sup_{\{f_t\}_{t=1}^T} \left\{ \sum_{t=1}^T f_t(x^\star) - \sum_{t=1}^T f_t(x_t) \right\},$$

where $x^\star = \operatorname{argmax}_{x \in \mathcal{X}} \sum_{t=1}^T f_t(x)$ is the (typically unknown) *best-in-hindsight* cache decision that can be selected only with access to future requests.[1] The regret measures the accumulated reward gap between the online decisions $\{x_t\}_t$ and benchmark $x^\star$. An algorithm is said to achieve sublinear regret when its average performance gap R_T/T vanishes as $T \to \infty$. In this context, recent works have proposed caching policies that offer $O(\sqrt{T})$ regret bound [11–17], which, in fact, is the optimal (as small as possible) achievable regret rate, see [18, Thm. 5.1], [11, Thm. 1].

Most of these regret-optimal algorithms have been designed for *continuous* caching, where it is assumed that each file is encoded and divided into a large number of small chunks such that storing them can be approximated by continuous variables [19]. In this case, the set of eligible caching states $\mathcal{X}$ is convex and hence one can readily apply standard OCO algorithms such as the Online Gradient Ascent (OGA). Albeit a practical assumption, there are settings where continuous caching cannot be used for practical reasons. Namely, keeping chunk meta-data consumes non-negligible storage; the coding operation is often computationally demanding; and the number of chunks might not be big enough to render continuous caching a good approximation. Therefore, we consider here the more realistic, and more challenging to solve, *discrete* caching problem. Indeed, in discrete caching the set $\mathcal{X}$ is naturally non-convex (containing binary file-caching decisions) and thus standard OCO policies cannot be employed. While first steps in the study of discrete caching, with equal-sized files, were recently made by [11, 12, 16]. In this paper, we extend their scope and design algorithms with substantially improved performance guarantees.

Namely, while regret minimization yields robust policies that learn under adversarial conditions, this framework receives the fair criticism that the policies have often suboptimal performance when the requests (cost functions, in general) are predictable, e.g., stationary. In such situations we would like the policy to gauge the predictability of requests, and optimize aggressively the cache. For instance, requests in services like Facebook are often amenable to accurate forecasts; while in YouTube and Netflix the viewers receive recommendations which can effectively serve as predictions for their forthcoming requests [20, 21]. Unfortunately, regret-based caching policies, such as [11–14, 16, 22], are *pessimistically* designed for the worst-case request sequence and cannot benefit from predictable requests. We tackle this shortcoming by designing a new suite of *optimistic* caching algorithms. An optimistic algorithm [23, 24] has access to a prediction of an unknown

---

[1]It is interesting to note that $x^\star$ caches the most frequent requests, which coincides with the limit behavior of LFU.

quality for the next-slot utility function. The ultimate goal is to achieve constant (independent of T) regret when the predictions are accurate, while maintaining the worst-case regret bounds when predictions fail. This best-of-two-worlds approach, we show theoretically and demonstrate numerically, brings significant performance gains to dynamic caching.

1.2 Methodology and Contributions

We study key variants of the discrete caching problem, namely the single cache with equal or unequal-sized files and the bipartite caching, and propose a suite of optimistic learning algorithms with different pros and cons.[2] Our first result demonstrates the best achievable regret in the setup we consider, which turns to be $R_T = \Omega(\sum_t \|\theta_t - \tilde{\theta}_t\|^{1/2})$, indicating a significant potential of obtaining a regret that scales with the predictor's error rather than the time horizon T (Sec. 3). We then proceed to propose variants of the seminal Follow-The-Regularized-Leader (FTRL) and Follow-the-Perturbed-Leader (FTPL) algorithms, which can be both viewed as *smoothing* techniques for stabilizing learning decisions (Sec. 2.2), whose regret match this lower bound up to constants. In detail, we expand the optimistic FTRL algorithm [15, 17, 24] that was designed for convex problems, to handle, through sampling, discrete (non-convex) decisions (Sec. 4). We prove this approach attains expected regret $O(\sqrt{T})$ for worst-case predictions and *zero-regret* for perfect predictions with an improved prefactor that does not depend on library size N. However, the OFTRL implementation can be hindered by an involved projection step that might be computationally expensive.[3] Thus, we develop a new optimistic FTPL algorithm that applies prediction-adaptive perturbations to achieve a similar regret bound with linear ($O(N)$) computation overhead (Sec. 5). The flip side is that its contains $O(poly\text{-}log(N))$ term.

We first derive results for equal-sized files, in line with all prior learning-based works for discrete caching [11, 16, 26, 27] or continuous caching [12, 14, 15]. Subsequently, we drop this assumption and study the single cache problem with different file sizes (Sec. 6). These first-of-their-kind regret-based algorithms require an elaborate point-wise approximation scheme for solving efficiently the NP-Hard Knapsack instance at each slot, while keeping the accumulated regret bound sublinear. To that end, we use the help of two rounding subroutines, DepRound [28] and Dantz [29]. We show that the proposed policies achieve $(1/2)$-approximate regret of $O(\sqrt{T})$ and zero-regret for adversarial and perfect predictions, respectively. We also extend the OFTRL analysis to the widely used bipartite network caching model [2, 4] (Sec. 7). In that case, we optimize jointly the discrete caching and routing decisions, to obtain a similar prediction-modulated performance.

In (Sec. 8), we change tack and incorporate the optimism through the celebrated Experts model. The caching system in this case is a *meta-learner* which receives caching advice from an *optimistic expert* that suggest to cache solely w.r.t. predicted requests, and from a *pessimistic expert* that ignores predictions. We propose a tailored OGD-based scheme that allows the meta-learner to adapt to predictions' accuracy (the performance of optimistic expert) in a way that achieves negative regret when that expert is reliable, and, again, maintain an $O(\sqrt{T})$ regret for unreliable predictions.

In summary, we provide a comprehensive toolbox of algorithms having different computation overheads and performance, hence enabling practitioners to select the best approach to their problem. Moreover, we include technical results that are of independent interest, such as the non-convex OFTPL algorithm with improved regret bounds; the approximate non-convex OFTRL algorithm for the Knapsack problem; and an analysis of OFTRL/OFTPL with a probabilistic prediction model.

---

[2]Optimistic learning was original proposed for problems with slowly-varying (hence, predictable) cost functions [23]; in caching, we note the additional motivation coming from the abundance of forecasting models, *e.g.,* by a Neural Network.
[3]In some cases the projection can be optimized, but in general it is $O(N^2)$ even for the non-weighted capped simplex [25].

Notation. We denote sets with calligraphic capital letters, e.g., $\mathcal{N} = \{1, 2, \ldots, N\}$; vectors with $x = (x_i, i \in \mathcal{N})$ where x_i is the ith component; and denote x_i^t the ith component of the time-indexed vector x_t. The shorthand notation $x_{1:t}$ is used for $\sum_{i=1}^{t} x_i$. Also, $\{x_t\}_{t=1}^{k}$ denotes the sequence of vectors $\{x_1, x_2, \ldots, x_k\}$, and we use the succinct version $\{x_t\}_T$ for $\{x_1, x_2, \ldots, x_T\}$. When clear from context, we often drop the notation of actions and denote the regret $R_T(\{x\}_T)$ simply with R_T.

2 BACKGROUND AND RELATED WORK

2.1 Caching and Learning

Research on caching optimization spans several decades and we refer the reader to survey [8] for an introduction to the recent developments in this area. A large body of works focus on offline policies which use the anticipated request pattern to proactively populate the caches with files that maximize the expected hits [1]. At the other extreme, dynamic caching solutions studied variants of the LFU/LRU policies [6, 7, 30, 31]; track the request distribution [32, 33] and optimize accordingly the caching [34]; employed reinforcement learning to adapt the caching decisions to requests [35–37]; and, more recently, applied online convex optimization towards enabling the policies to handle unknown (adversarial) request patterns [11–16, 22]. These latter works assume that the files can be fetched dynamically at each slot to optimize the cache configuration, as opposed to works such as [26, 27] which study pure eviction policies.

The interplay between predictions and caching has attracted attention from both machine learning and networking communities. The studies in [38–40] formulated the joint caching and recommendation problem, considering static models, and assuming full knowledge of requests and the users' propensity to follow recommendations (i.e., they place assumption on the prediction accuracy). On the other hand, [26, 27] presented a mechanism agnostic to requests that uses untrusted predictions to achieve *competitive-ratio* guarantees. Their approach was generalized to metrical task systems by [41] and improved with nearly lower-bound matching for the competitive ratios in [42]. However, as proved in [43], algorithms that ensure constant competitive-ratios do not necessarily guarantee sub-linear regret, which is the performance criterion we employ here following the recent regret-based caching research [11–16, 22]. We note that all above works consider files with equal size, while we extend the framework to the general scenario of *unequally-sized* files. In addition, none of the above works studies discrete caching with predictions. Finally, it is worth stressing that employing predictions for improving the performance of communication/computing systems is not a new idea: predictions have been incorporated in stochastic optimization [44, 45] which assume the requests and system perturbations are *stationary*; and in online learning [46, 47] which *do not adapt* to predictions' accuracy (considered known). Here, we make *no assumptions* on the predictions' quality, which can be even adversarial.

2.2 Adaptive Smoothing

In contrast to the above studies, our optimistic learning approach is based on *adaptive smoothing*. Abernethy *et. al.* [48] introduced a unified view of FTRL and FTPL as techniques to add smoothing, through *regularization* or *perturbation*, to a non-smooth potential function. This perspective is useful to our work since we leverage both ideas. Namely, let us define: $\Phi(\theta) \triangleq \max_{x \in \mathcal{X}} \langle x, \theta \rangle$, and consider the potential function $\Phi(\Theta_t)$, where $\Theta_t = \theta_{1:t}$ is the vector of aggregated gradients (file requests). An intuitive strategy is to choose the action that maximizes the rewards seen so far:

$$x_t = \operatorname*{argmax}_{x \in \mathcal{X}} \langle \Theta_{t-1}, x \rangle = \nabla \Phi(\Theta_{t-1}), \tag{1}$$

which is known as Follow The Leader (FTL) and is optimal when the utility functions are samples from a stationary statistical distribution. In contrast, FTL has linear regret in the adversarial setting

[49, 50], since successive gradients of non-smooth functions can be arbitrary far to each other, thus leading to unstable actions. [48] proposed to stabilize the learner actions by *smoothing* the potential function, and selecting actions based on the *smoothed* potential $\nabla\tilde{\Phi}(\theta)$. In FTRL, the smoothing is achieved by adding a strongly convex function to the potential, i.e.,[4] $x_t = \text{argmax}_{x\in\text{conv}(\mathcal{X})}\langle x, \Theta_t\rangle - r_{1:t}(x)$ where $r_t(x)$ is a σ_t-strongly convex regularizer. This framework generalizes the Online Gradient Ascent (OGA) and the Exponentiated Weights (EG) algorithms, which were employed for the caching problem in [14] and [12] respectively[5]. As for FTPL, the smoothing is done by adding perturbation to the accumulated cost parameter of the potential. And the actions are decided by[6] $x_t = \arg\max_{x\in\mathcal{X}}\langle x, \Theta_t + \eta_t\gamma\rangle$, where $\gamma\sim\mathcal{N}(0,\mathbf{1})$ and η_t is a scaling factor that controls the smoothing. FTPL was shown to provide optimal regret guarantees for the discrete caching problem in [11]. Computationally efficiency is also a notable feature for FTPL updates as it requires an ordering operation instead of projection.

We propose to modulate the regularization σ_t and perturbation γ_t parameters with the predictions quality. Intuitively, accurate predictions should lead to smaller regularization/perturbation and hence less smoothing, enabling the learner to align its decisions to the predictions. On the other hand, inaccurate predictions induce more smoothing, which alleviates their effects on the decisions. We show that careful tuning of these smoothing parameters leads to regret bounds that interpolate between $R_T \leq 0$, and $R_T \leq O(\sqrt{T})$. Nonetheless, these two algorithms have considerable differences in terms of computational complexity and constants in the bounds, which are discussed in detail.

2.3 Optimistic Learning

For *regret* minimization with predictions, [52] used predictions θ_t for the gradient $\tilde{\theta}_t = \nabla f_t(x_t)$ with guaranteed correlation $\langle\tilde{\theta}_t, \theta_t\rangle \geq \alpha\|\theta_t\|^2$ to improve the regret . In [53], this assumption was relaxed to allow predictions to fail the correlation condition at some steps, obtaining bounds that interpolate between $O(\log(T))$ and $O(\sqrt{T})$; while this idea was extended to multiple predictors in [54]. A different line of works [23], [24] use adaptive regularizers and define the t-slot prediction errors $\|\theta_t - \tilde{\theta}_t\|_2$ to obtain $O([\sum_t \|\theta_t - \tilde{\theta}_t\|^2]^{1/2})$ regret bounds. Specifically, OFTRL versions have been proposed in [24] and recently used in [55] for problems with budget constraints, while [15, 17] tailored these ideas to *continuous* caching. The problem of discrete caching is fundamentally different. Through a careful analysis, we manage to reuse these results after relaxing the cache integrality constraints, and then employ a randomized rounding technique that recovers the same prediction-modulated regret in expectation. The regret bounds have the desirable property of being *dimension-free*. Nonetheless, we proceed to remark that OFTRL can have a computational bottleneck due to involving a projection step, which can be avoided in FTPL.

Optimistic versions of FTPL were recently investigated in [56] and [57]. In [56], the regret bound grows polynomially w.r.t. the decision set dimension. In the caching problem, this would imply a highly-problematic polynomial growth of the regret w.r.t. the typically huge library size N. The dependence of the regret on the dimension was improved in [57], but it still remains linear. On the contrary, our proposed OFTPL exploits the structure of the decision set and utilizes adaptive perturbation to obtain a regret bound that depends on dimension only by $O(\log(N)^{1/4})$, is order-optimal (based on the achievable lower bound), returns zero-regret for perfect predictions, does not require knowing the time-horizon T, nor the prediction errors. None of these desirable features

---

[4]While the maximization requires that x_t to be in the convex hull of $\mathcal{X}$, feasibility can be recovered via appropriate rounding.
[5]We note that these papers present their algorithms as instances of a similar framework to FTRL called Online Mirror Descent (OMD). Nonetheless, there exist equivalence results between these two frameworks (see [51, Sec. 6.1]) for specific choices of the *mirror-map* (in OMD), or equivalently the regularizer (in FTRL).
[6]In this case the gradient of the smoothed potential is in fact the expectation $\nabla\Phi(\Theta_t + \eta_t\gamma) = \mathbb{E}_\gamma[x_t]$

is available in these prior works. We kindly refer the reader to the table in Appendix. A.1 for an overview of the presented algorithms in the context of the most related literature.

3   ACHIEVABLE REGRET FOR CACHING WITH A PREDICTOR

We first introduce a lower bound for the regret of any online caching policy π, working with a cache of capacity C, and has access to an *untrusted* and potentially *adversarial* prediction oracle. In general, the predictions refer to the next function $\tilde{f}_t(\cdot)$. However, since most OCO algorithms learn based on the observed gradients, it suffices to have predictions $\tilde{\theta}_t = \nabla \tilde{f}_t(x_t)$. And for caching, this coincides with a prediction for the next request[7]. Now, unlike all prior works in optimistic learning [23, 24, 53], we adopt here the more general *probabilistic* prediction model where $\tilde{\theta}_t$ is not necessarily a one-hot vector (as the actual θ_t), but a probability distribution over the library. Thus, each $\tilde{\theta}_t$ is drawn from the N-dimensional probability simplex Δ_N. This more general approach is rather intuitive as the forecasting models (e.g., a Neural Network) typically yield probabilistic inferences. It also enhances the performance of our optimistic algorithms and allows an efficient training of the forecaster using a convex loss function (please see Appendix A.7.1 for examples and justification). It does require, however, a more elaborate technical analysis, especially for the case of OFTPL. In this setup, we have the following lower bound:

THEOREM 1. *The regret R_T of any online caching policy π, with access to $\{\tilde{\theta}_t\}_t$, is lower bounded as*

$$\mathbb{E}\left[R_T\right] \geq \sqrt{\frac{C}{2\pi}} \sqrt{\sum_{t=1}^{T} ||\theta_t - \tilde{\theta}_t||_2^2} - \Theta\left(\frac{1}{\sqrt{T}}\right).$$

PROOF. To prove the lower bound, we show the existence of a request and prediction sequence under which the regret is guaranteed to be larger than the stated bound regardless of the online policy π. For that, we use the standard probabilistic method [58] with an appropriately constructed random file request and prediction sequence as detailed below.

Denote by ξ_t and $\tilde{\xi}_t$ the random variables representing the requested file (θ_t) and its prediction ($\tilde{\theta}_t$) at time t, respectively. Denote by $\{X_t^\pi\}_{t \geq 1}$ the random variables representing the action of any policy π. We use a setup where $N \geq 2C$ and consider an ensemble of caching problems (i.e., request and prediction sequences) where at each slot t, the requested file ξ_t is chosen independently and uniformly at random from the library $\mathcal{N}$. The prediction $\tilde{\xi}_t$ are also chosen independently and uniformly at random from the probability simplex Δ_N. Specifically, we let

$$\{\tilde{\xi}_t\}_t \overset{\text{i.i.d.}}{\sim} \text{DIRICHLET}(\lambda_1, \ldots, \lambda_n, \ldots \lambda_N) \quad \text{with} \quad \lambda_n = 1, \forall n \in \mathcal{N}.$$

Hence, the expected reward obtained by any caching policy π on any slot t, conditional on the information available to the policy can be bounded as

$$\mathbb{E}\left[\langle \xi_t, X_t^\pi \rangle | \{\tilde{\xi}_\tau\}_{\tau=1}^{t}, \{\xi_\tau\}_{\tau=1}^{t-1}\right] \overset{(a)}{=} \mathbb{E}\mathbb{E}\left[\langle \xi_t, X_t^\pi \rangle | \{\tilde{\xi}_\tau\}_{\tau=1}^{t}, \{\xi_\tau\}_{\tau=1}^{t-1}, X_t^\pi\right]$$

$$\overset{(b)}{=} \frac{1}{2C}\mathbb{E}\left[\langle \mathbf{1}, X_t^\pi \rangle | \{\tilde{\xi}_\tau\}_{\tau=1}^{t}, \{\xi_\tau\}_{\tau=1}^{t-1}, X_t^\pi\right] \overset{(c)}{\leq} \frac{1}{2},$$

where (a) follows from the tower property of expectations, (b) from the fact $\xi_t \perp \left(\{\tilde{\xi}_\tau\}_{\tau=1}^{t}, \{\xi_\tau\}_{\tau=1}^{t-1}, X_t^\pi\right)$ and hence $\mathbb{E}(\xi_t|\{\tilde{\xi}_\tau\}_{\tau=1}^{t}, \{\xi_\tau\}_{\tau=1}^{t-1}, X_t^\pi) = \mathbb{E}(\xi_t) = \frac{1}{2C}\mathbf{1}_{N \times 1}$. Finally (c) since $\langle \mathbf{1}, X_t^\pi \rangle \leq C$, which holds because of the cache capacity constraint. Taking expectation of the above bound, we have

---

[7]In fact this model can be readily generalized to other linear utilities beyond cache-hits, so as to incorporate e.g., file-specific caching gains, time-varying network conditions, and so on; see similar models in [14, 17].

$\mathbb{E}[\langle \xi_t, X_t^\pi \rangle] \le \frac{1}{2}$. Hence, using the linearity of expectations, the expected value of the cumulative hits up to slot T under any policy π is upper bounded as $\mathbb{E}\big[\sum_{t=1}^{T} \langle \xi_t, X_t^\pi \rangle\big] \le \frac{T}{2}$.

Now we compute a lower bound to the expected number of cumulative hits achieved by the best-in-hindsight fixed cache configuration $X_T^\star$. Similar to [11], we identify the problem with the classic setup of balls (requests) into bins (files). In this framework, it follows that the offline benchmark achieves cumulative hits which are equal to the total number of balls into the most-loaded C bins when T balls are thrown uniformly at random into $N = 2C$ bins. Hence, from [11, Lemma 1]:

$$\mathbb{E}\big[\sum_{t=1}^{T} \langle \xi_t, X_T^\star \rangle\big] \ge \frac{T}{2} + \sqrt{\frac{CT}{2\pi}} - \Theta\left(\frac{1}{\sqrt{T}}\right).$$

Hence, the expected regret achieved by any policy in the optimistic set up is lower bounded as

$$\mathbb{E}[R_T] = \mathbb{E}\left(\sum_{t=1}^{T} \langle \xi_t, X_T^\star \rangle - \sum_{t=1}^{T} \langle \xi_t, X_t^\pi \rangle\right) \ge \sqrt{\frac{CT}{2\pi}} - \Theta\left(\frac{1}{\sqrt{T}}\right). \tag{2}$$

Finally, we evaluate the expected value of the quantity $M_T \triangleq \sum_{t=1}^{T} \|\tilde{\xi}_t - \xi_t\|_2^2$ as follows.

$$\mathbb{E}[M_T] = \mathbb{E}\left[\sum_{t=1}^{T} \|\tilde{\xi}_t - \xi_t\|_2^2\right] \stackrel{(a)}{=} T\,\mathbb{E}\left[\|\tilde{\xi}_1 - \xi_1\|_2^2\right] \stackrel{(b)}{=} TN\,\mathbb{E}\left[(\tilde{\xi}_1^1 - \xi_1^1)^2\right]$$

$$= TN\,\mathbb{E}\left[(\tilde{\xi}_1^1)^2 - 2\tilde{\xi}_1^1\xi_1^1 + (\xi_1^1)^2\right] \stackrel{(c)}{=} TN\left[\text{Var}(\tilde{\xi}_{1,1}) + (\mathbb{E}(\tilde{\xi}_{1,1}))^2 - 2\mathbb{E}(\tilde{\xi}_{1,1})\mathbb{E}(\xi_{1,1}) + \mathbb{E}(\xi_{1,1}^2)\right]$$

$$\stackrel{(d)}{=} TN\left[\frac{(N-1)}{N^2(N+1)} + \frac{1}{N^2} - \frac{2}{N^2} + \frac{1}{N}\right] = T\left(1 - \frac{2}{N(N+1)}\right) \le T,$$

where (a) follows from the i.i.d. assumption of the random vectors at each t, (b) from the i.i.d assumption of each component of vectors $\tilde{\xi}_1$ and ξ_1, (c) from $\xi_t \perp \tilde{\xi}_t$, and (d) from standard results on Dirichlet distribution. Combining the above bound with (2), we have by Jensen's inequality

$$\mathbb{E}[R_T] \ge \sqrt{\frac{C\mathbb{E}[M_T]}{2\pi}} - \Theta\left(\frac{1}{\sqrt{T}}\right) \ge \mathbb{E}\left[\sqrt{\frac{CM_T}{2\pi}} - \Theta\left(\frac{1}{\sqrt{T}}\right)\right] \qquad i.e.,$$

$$\mathbb{E}\left[R_T - \sqrt{\frac{C\sum_{t=1}^{T} \|\tilde{\xi}_t - \xi_t\|_2^2}{2\pi}}\right] \ge -\Theta\left(\frac{1}{\sqrt{T}}\right)$$

From the above inequality, the result now follows from the standard probabilistic arguments. □

We will see that the proposed optimistic algorithms in Sections 4 and 5 attain this bound within an absolute and a poly-logarithmic factor respectively.

4 CACHING THROUGH OPTIMISTIC REGULARIZATION (OFTRL-CACHE)

The first algorithm we propose is based on OFTRL. The gist of our approach is that we use OFTRL to obtain $\hat{x}_t \in \text{conv}(\mathcal{X})$, $\forall t$, and then apply Madow's sampling scheme [59] to recover integral caching vectors $x_t \in \mathcal{X}$, $\forall t$, which satisfy the hard capacity constraint. In other words, we define:

$$\mathcal{X} = \left\{x \in \{0,1\}^N \,\middle|\, \sum_{i=1}^{N} x_i \le C\right\},$$

Algorithm 1: Optimistic Follow The Regularized Leader (`OFTRL-Cache`)

---

1 **Input**: $\sigma = 1/\sqrt{C}$, $\delta_1 = \|\theta_1 - \tilde{\theta}_1\|_2^2$, $\sigma_1 = \sigma\sqrt{\delta_1}$, $x_1 \in \mathcal{X}$

2 **Output**: $\{x_t \in \mathcal{X}\}_T$ // *Feasible discrete caching vector at each slot*

3 **for** $t = 2, 3 \dots$ **do**

4      $\tilde{\theta}_t \leftarrow$ prediction   // *Obtain request prediction for slot t*

5      $\hat{x}_t = \text{argmax}_{x \in \text{conv}(\mathcal{X})} \left\{ -r_{1:t-1}(x) + \langle x, \Theta_{t-1} + \tilde{\theta}_t \rangle \right\}$ // *Update the continuous cache vector*

6      $x_t \leftarrow MadowSample(\hat{x}_t)$ // *Obtain the discrete cache vector using Algorithm 7*

7      $\Theta_t = \Theta_{t-1} + \theta_t$ // *Receive t-slot request and update total gradient*

8      $\sigma_t = \sigma\left(\sqrt{\delta_{1:t}} - \sqrt{\delta_{1:t-1}}\right)$ // *Update the regularization parameter*

    end

---

where $\mathcal{N}$ is the set of unit-sized files (library) and C is the cache capacity (in file units); and $x_i = 1$ decides to cache file $i \in \mathcal{N}$. Interestingly, despite having to operate on this non-convex set, this approach yields *in expectation* the same regret bounds as OFTRL for continuous caching [17].

Let us define the prediction error at slot t as $\delta_t \triangleq \|\theta_t - \tilde{\theta}_t\|_2^2$, and introduce the proximal σ_t-strongly convex regularizer w.r.t. the Euclidean ℓ_2 norm:

$$r_t(x) = \frac{\sigma_t}{2}\|x - x_t\|_2^2. \tag{3}$$

Following [15], we define parameters $\{\sigma_t\}_t$ using the accumulated prediction errors, namely:

$$\sigma_1 = \sigma\sqrt{\delta_1}, \qquad \sigma_t = \sigma\left(\sqrt{\delta_{1:t}} - \sqrt{\delta_{1:t-1}}\right) \quad \forall t \geq 2, \quad \text{with} \quad \sigma = 1/\sqrt{C}.$$

The basic OFTRL update stems from using these regularizers in the FTRL update formula. Namely, at each slot t we update the cache to maximize the aggregated utility. This maximization is *regularized* through a term (the above-defined regularizers) that depends predictor's accuracy. The detailed steps are summarized in Algorithm 1. In the first iteration we draw randomly a feasible caching vector x_1 and observe the prediction error $\delta_1 = \|\theta_1 - \tilde{\theta}_1\|_2^2$. In each iteration we need to solve a strongly convex program (line 5) which returns the continuous caching vector $\hat{x}_t$, that is transformed to a feasible discrete x_t (line 6) using Madow's Sampling (see Appendix A.2). The algorithm notes the new gradient vector, by simply observing the next request[8], and updates the accumulated gradient Θ_t (line 7). The regret guarantee of Algorithm 1 is described next.

THEOREM 2. *Algorithm 1 ensures, for any time horizon T, the expected regret bound:*

$$\mathbb{E}[R_T] \leq 2\sqrt{C}\sqrt{\sum_{t=1}^{T} \|\theta_t - \tilde{\theta}_t\|_2^2}$$

PROOF. We define first the regret w.r.t. the continuous actions $\{\hat{x}_t\}_T$ as $\hat{R}_T \triangleq \langle \Theta_T, x^\star \rangle - \sum_{t=1}^{T}\langle \theta_t, \hat{x}_t \rangle$, where $x^\star$ is the optimal-in-hindsight caching vector[9]. We also define the scaled Euclidean ℓ_2 norm $\|\cdot\|_{(t)} = \sqrt{\sigma_{1:t}}\|\cdot\|_2$ so that $r_{1:t}$ is 1-strongly convex w.r.t $\|\cdot\|_{(t)}$, and note that its dual norm is $\|\cdot\|_{(t),\star} = \frac{1}{\sqrt{\sigma_{1:t}}}\|\cdot\|_2$. Our starting point is [15, Lem. 1], which we restate below:

LEMMA 1. *Let $r_{1:t}$ be a 1-strongly convex w.r.t. a norm $\|\cdot\|_{(t)}$. Then, the OFTRL iterates produced by line 5 in Algorithm 1 guarantee the bound $\hat{R}_T \leq r_{1:T}(x^\star) + \frac{1}{2}\sum_{t=1}^{T}\|\theta_t - \tilde{\theta}_t\|_{(t),\star}^2$.*

---

[8]For modeling convenience, we define the time slots to be the (non-uniform) time intervals that receive only one request. Our analysis can be readily extended to a bounded number of requests per slot.

[9]This benchmark remains unchanged if we switch from the continuous to the discrete space.

Now, we first get a deterministic regret bound on $\hat{x}_t$. Assuming that[10] $C \in (0, N/2]$, we can bound the ℓ_2 diameter of the caching set as $\|x^\star - x_t\|_2 \le \sqrt{2C}, \forall x^\star, x_t \in \text{conv}(X)$. Thus, we can upper-bound the regularizers in (3), replace in the above Lemma and telescope to get:

$$\hat{R}_T \le \sigma_{1:T} C + \frac{1}{2} \sum_{t=1}^{T} \frac{\delta_t}{\sigma_{1:t}}. \tag{4}$$

Observing that the sum $\sigma_{1:t}$ telescopes to $\sigma\sqrt{\delta_{1:t}}$, we can substitute it in (4) and use the standard identity [18, Lem. 4.13] to bound the second term via $\sum_{t=1}^{T} \delta_t/\sqrt{\delta_{1:t}} \le 2\sqrt{\delta_{1:t}}$. Therefore, we obtain: $\hat{R}_T \le \sigma\sqrt{\delta_{1:T}} C + \frac{1}{\sigma}\sqrt{\delta_{1:T}}$, and by setting $\sigma = 1/\sqrt{C}$ we can eventually write:

$$\hat{R}_T \le 2\sqrt{C}\sqrt{\sum_{t=1}^{T} \|\theta_t - \tilde{\theta}_t\|_2^2}. \tag{5}$$

The last step requires Madow's sampling (line 6). By construction, the routine selects C files and hence returns a feasible integral caching vector x_t (or, *sampled vector*). In addition, each item is included in the sampled vector with a probability based on the continuous $\hat{x}_t$. Namely, it holds $\text{Pr}(x_i^t = 1) = \pi_i - \pi_{i-1} = \hat{x}_i^t$, where the auxiliary parameter π_i aggregates the (interpreted as) probabilities for caching the first i files, i.e., $\pi_i = \sum_{k=0}^{i} \hat{x}_k$. Since each x_i^t is binary, it holds $\mathbb{E}[x_t] = \text{Pr}(x_i^t = 1) = \hat{x}_t$. The result follows by using (5) and observing:

$$\mathbb{E}\left[R_T(\{x\}_T)\right] = \langle \Theta_T, x^\star \rangle - \mathbb{E}\left[\sum_{t=1}^{T} \langle \theta_t, x_t \rangle\right] = \hat{R}_T.$$

$\square$

Discussion. The bound in Theorem 2 ensures the desirable prediction-based modulation of the algorithm's performance, as the achieved regret *shrinks* with the prediction quality. If all predictions are accurate, we get $R_T \le 0$; when all predictions fail, we get $R_T \le 2\sqrt{2CT}$. That is, in the worst scenario (e.g., when the predictions are created by an adversary) the regret bound is worse by a constant factor of $\sqrt{2}$ compared to the FTRL algorithm that does not use predictions [51, Sec. 3.4]), and ~ 5 compared to the lower bound derived in Sec. 3. Moreover, due to selecting an ℓ_2 regularizer, the bounds are dimension-free and do not depend on the library size N. This is particularly important since in caching problems oftentimes the library size is an even bigger concern than the time horizon. Finally, note that the algorithm does not need to know the horizon T beforehand. The drawback of this optimistic caching approach is the computational complexity of the iteration (line 5) which involves a projection operation. While ℓ_2 projections have received attention [10, Sec. 7], they can hamper the scalability of the algorithm under certain conditions[11]. In the following section we show how perturbation-based smoothing can avoid the projection step.

5 CACHING THROUGH OPTIMISTIC PERTURBATIONS (OFTPL-CACHE)

We propose next a new OFTPL algorithm that significantly improves previous OFTPL proposals [56, 57], both in terms of their bounds and implementation, and as such is of independent interest with potential applications that extend beyond caching to other $k-$set structured problems such as those discussed in [60].

---

[10]N is typically order of magnitudes higher than C. In the cases where this does not hold the current analysis is still valid but can be improved by using the tighter diameter $\sqrt{2(N-C)}$.

[11]For instance, this can be a bottleneck if the library size is extremely large, while the slot duration is very short and the available computation power limited.

Algorithm 2: Optimistic Follow The Perturbed Leader (`OFTPL-Cache`)

---

1 **Input**: $\eta_1 = 0$, $y_1 \in \mathcal{X}$
2 **Output**: $\{y_t \in \mathcal{X}\}_T$ // *Feasible discrete caching vector at each slot*
3 $\gamma \overset{iid}{\sim} \mathcal{N}(0, \mathbf{1}_{N \times 1})$ // *Sample a perturbation vector*
4 **for** $t = 2, 3, \dots$ **do**
5 $\quad$ $\tilde{\theta}_t \leftarrow$ prediction   // *Obtain request prediction for slot t*
6 $\quad$ $\eta_t = \frac{1.3}{\sqrt{C}} \left(\frac{1}{\ln(Ne/C)} \right)^{\frac{1}{4}} \sqrt{\sum_{\tau=1}^{t-1} \|\theta_\tau - \tilde{\theta}_\tau\|_1^2}$ // *Update the perturbation parameter*
7 $\quad$ $y_t = \operatorname{argmax}_{y \in \mathcal{X}} \langle y, \Theta_{t-1} + \tilde{\theta}_t + \eta_t \gamma \rangle$ // *Update the discrete cache vector*
8 $\quad$ $\Theta_t = \Theta_{t-1} + \theta_t$ // *Receive the request for slot t and update total gradient*
end

Following the discussion in Sec. 2, we remind the reader that the FTPL actions are derived by solving in each slot t a linear program (LP) with a parameterized perturbed cumulative utility vector, $\Theta_{t-1} + \eta_t \gamma$, where $\eta_t \in \mathbb{R}_+$ is the perturbation parameter. In order to obtain the optimistic FTPL variant we introduce two twists: *(i)* the prediction for the next-slot utility $\tilde{\theta}_t$ is added to the cumulative utility; and *(ii)* the perturbation parameter η_t is scaled according to the accumulated prediction error. Interestingly, due to the structure of the decision set $\mathcal{X}$, the LP solution reduces to identifying the C files with the highest coefficients. This step can be efficiently implemented in deterministic linear ($O(N)$) time using, e.g., the Median-of-Medians algorithm [61]. The steps of the proposed scheme are presented in Algorithm 2, where we denote the t-slot OFTPL decisions with $y_t \in \mathcal{X}$. The following theorem characterizes the performance of this new OFTPL algorithm.

THEOREM 3. *Algorithm 2 ensures, for any time horizon T, the expected regret bound:*

$$\mathbb{E}_\gamma[R_T] \leq 3.68\sqrt{C} \left(\ln \frac{Ne}{C} \right)^{1/4} \sqrt{\sum_{t=1}^{T} \|\theta_t - \tilde{\theta}_t\|_1^2}.$$

PROOF. We consider the following *baseline* potential function $\Phi(\theta) \triangleq \max_{y \in \mathcal{X}} \langle y, \theta \rangle$, which is a sub-linear function[12]. The associated *Gaussian smoothed* potential function for each t, is defined as:

$$\Phi_t(\theta) \triangleq \underset{\gamma \sim \mathcal{N}(0,I)}{\mathbb{E}} \left[\max_{y \in \mathcal{X}} \langle y, \theta + \eta_t \gamma \rangle \right] = \underset{\gamma}{\mathbb{E}} \left[\Phi(\theta + \eta_t \gamma) \right]$$

Clearly, $\Phi_t(\theta)$ is convex in θ. Recall that the cumulative file request vector is defined as $\Theta_t = \Theta_{t-1} + \theta_t$. A Taylor expansion of $\Phi_t(\cdot)$ around the point $\Theta_{t-1} + \tilde{\theta}_t$, evaluated at Θ_t, with a second order remainder is:

$$\Phi_t(\Theta_t) = \Phi_t(\Theta_{t-1} + \tilde{\theta}_t) + \langle \nabla \Phi_t(\Theta_{t-1} + \tilde{\theta}_t), \theta_t - \tilde{\theta}_t \rangle + \frac{1}{2} \langle \theta_t - \tilde{\theta}_t, \nabla^2 \Phi_t(\hat{\theta}_t)(\theta_t - \tilde{\theta}_t) \rangle, \quad (6)$$

where $\hat{\theta}_t$ is a point on the line segment connecting Θ_t and $\Theta_{t-1} + \tilde{\theta}_t$. From the convexity of $\Phi_t(\cdot)$:

$$\Phi_t(\Theta_{t-1} + \tilde{\theta}_t) \leq \Phi_t(\Theta_{t-1}) + \langle \nabla \Phi_t(\Theta_{t-1} + \tilde{\theta}_t), \tilde{\theta}_t \rangle. \quad (7)$$

From (6) and (7), we can eventually write:

$$\Phi_t(\Theta_t) \leq \Phi_t(\Theta_{t-1}) + \langle \nabla \Phi_t(\Theta_{t-1} + \tilde{\theta}_t), \theta_t \rangle + \frac{1}{2} \langle \theta_t - \tilde{\theta}_t, \nabla^2 \Phi_t(\hat{\theta}_t)(\theta_t - \tilde{\theta}_t) \rangle. \quad (8)$$

---

[12] A function f is sub-linear if it is sub-additive (i.e., $f(a) + f(b) \geq f(a+b)$, which implies $f(a) - f(b) \leq f(a-b)$), and positive homogeneous (i.e., $f(\lambda a) = \lambda f(a), \lambda > 0$).

Now, note that it holds $\nabla \Phi_t(\Theta_{t-1} + \tilde{\theta}_t) =$

$$= \nabla \mathop{\mathbb{E}}_{\gamma} \left[\Phi_t(\Theta_{t-1} + \tilde{\theta}_t) \right] \overset{(a)}{=} \mathop{\mathbb{E}}_{\gamma} \left[\nabla \Phi_t(\Theta_{t-1} + \tilde{\theta}_t) \right] = \mathop{\mathbb{E}}_{\gamma} \left[\arg\max_{y \in \mathcal{X}} \langle y, \Theta_{t-1} + \tilde{\theta}_t + \eta_t \gamma \rangle \right] = \mathop{\mathbb{E}}_{\gamma} [y_t],$$

where (a) stems from [62, Prop. 2.2]. Thus, (8) can be written as:

$$\Phi_t(\Theta_t) \leq \Phi_t(\Theta_{t-1}) + \mathop{\mathbb{E}}_{\gamma} [\langle \theta_t, y_t \rangle] + \frac{1}{2} \langle \theta_t - \tilde{\theta}_t, \nabla^2 \Phi_t(\hat{\theta}_t)(\theta_t - \tilde{\theta}_t) \rangle.$$

Subtracting $\Phi_{t-1}(\Theta_{t-1})$ from both sides and telescoping over T and setting $\eta_0 = 0$, we get:

$$\Phi_T(\Theta_T) \leq \sum_{t=1}^{T} \left(\Phi_t(\Theta_{t-1}) - \Phi_{t-1}(\Theta_{t-1}) + \mathop{\mathbb{E}}_{\gamma} [\langle \theta_t, y_t \rangle] + \frac{1}{2} \langle \theta_t - \tilde{\theta}_t, \nabla^2 \Phi_t(\hat{\theta}_t)(\theta_t - \tilde{\theta}_t) \rangle \right).$$

Then, by Jensen's inequality: $\max_{y \in \mathcal{X}} \mathbb{E}_{\gamma} [\langle y, \Theta_T + \eta_T \gamma \rangle] = \max_{y \in \mathcal{X}} \langle y, \Theta_T \rangle = \Phi(\Theta_T) \leq \Phi_T(\Theta_T)$, and writing the last term as the norm of the vector $(\theta_t - \tilde{\theta}_t)$ induced by the symmetric positive semidefinite matrix $\nabla^2 \Phi_t(\hat{\theta}_t) \triangleq H_t$, we get the following upper bound of the regret:

$$R_T \leq \Phi(\Theta_T) - \sum_{t=1}^{T} \mathop{\mathbb{E}}_{\gamma} [\langle \theta_t, y_t \rangle] \leq \sum_{t=1}^{T} \left(\Phi_t(\Theta_{t-1}) - \Phi_{t-1}(\Theta_{t-1}) + \frac{1}{2} \|\theta_t - \tilde{\theta}_t\|_{H_t} \right). \qquad (9)$$

We now bound the first term in the RHS of inequality (9):

$$\sum_{t=1}^{T} \Phi_t(\Theta_{t-1}) - \Phi_{t-1}(\Theta_{t-1}) = \sum_{t=1}^{T} \mathop{\mathbb{E}}_{\gamma} \left[\Phi(\Theta_{t-1} + \eta_t \gamma) - \Phi(\Theta_{t-1} + \eta_{t-1} \gamma) \right] \overset{(a)}{\leq} \sum_{t=1}^{T} \mathop{\mathbb{E}}_{\gamma} \left[\Phi((\eta_t - \eta_{t-1})\gamma) \right]$$

$$\overset{(b)}{\leq} \sum_{t=1}^{T} (\eta_t - \eta_{t-1}) \mathop{\mathbb{E}}_{\gamma} [\Phi(\gamma)] \leq \eta_T \mathop{\mathbb{E}}_{\gamma} \left[\max_y \langle y, \gamma \rangle \right] \overset{(c)}{\leq} \eta_T \sqrt{2C \ln \binom{N}{C}} \overset{(d)}{\leq} \eta_T C \sqrt{2 \ln(Ne/C)}, \qquad (10)$$

where inequalities (a) and (b) follow from the sub-linearity of the potential function; (c) from Massart's lemma which gives an upper bound the expected sum of the top C elements in a Gaussian random vector (e.g., [60, Lem. 9]); and finally (d) is due to $\binom{N}{C} \leq (\frac{Ne}{C})^C$.

We now upper bound the second term in the RHS of (9). From [60, Eqn. (4)], the $(i,j)^{\text{th}}$ entry of the Hessian matrix is given by $H_{i,j}^t = \frac{1}{\eta_t} \mathbb{E}[\tilde{y}(\hat{\theta}_t + \eta_t \gamma)_i \gamma_j]$, where $\tilde{y}(\cdot) = \arg\max_{y \in \mathcal{X}} \langle y, \cdot \rangle$. Hence, we have the following bound on the absolute value of each entry:

$$|H_{i,j}^t| = \frac{1}{\eta_t} \left| \mathop{\mathbb{E}}_{\gamma} \left[\tilde{y}(\hat{\theta} + \eta_t \gamma)_i \gamma_j \right] \right| \leq \mathop{\mathbb{E}}_{\gamma} \left[|\tilde{y}(\hat{\theta} + \eta_t \gamma)_i| \, |\gamma_j| \right] \leq \frac{1}{\eta_t} \mathop{\mathbb{E}}_{\gamma} [|\gamma_i|] \leq \frac{1}{\eta_t} \sqrt{\frac{2}{\pi}}, \qquad (11)$$

where the first inequality follows from Jensen's inequality, the second holds since $\tilde{y}_i = \{0, 1\}$; and the last one is a property of Gaussian r.v.s. Thus each of the quadratic forms on the RHS of Eqn. (9) can be bounded as follows:

$$\|\theta_t - \tilde{\theta}_t\|_{H_t} = \langle \theta_t - \tilde{\theta}_t, H_t(\theta_t - \tilde{\theta}_t) \rangle = \sum_{i,j} (\theta_{t,i} - \tilde{\theta}_{t,i}) H_{ij}^t (\theta_{t,j} - \tilde{\theta}_{t,j})$$

$$\overset{(a)}{\leq} \sum_{i,j} |(\theta_{t,i} - \tilde{\theta}_{t,i})| |H_{ij}^t| |(\theta_{t,j} - \tilde{\theta}_{t,j})| \overset{(b)}{\leq} \frac{1}{\eta_t} \sqrt{\frac{2}{\pi}} (\sum_i |(\theta_{t,i} - \tilde{\theta}_{t,i})|)^2 = \frac{1}{\eta_t} \sqrt{\frac{2}{\pi}} \|\theta_t - \tilde{\theta}_t\|_1^2. \qquad (12)$$

where (a) follows from the triangle inequality and (b) from the bound (11).

Another way to bound $\|\theta_t - \tilde{\theta}_t\|_{H_t}$, which will be useful later, starts from (6) to get:

$$\frac{1}{2}\|\theta_t - \tilde{\theta}_t\|_{H_t} = \Phi_t(\Theta_t) - \Phi_t(\Theta_{t-1} + \tilde{\theta}_t) - \langle \nabla \Phi_t(\Theta_{t-1} + \tilde{\theta}_t), \theta_t - \tilde{\theta}_t \rangle$$

$$= \mathbb{E}_{\gamma} \Big[\Phi(\Theta_t + \eta_t \gamma) - \Phi(\Theta_{t-1} + \tilde{\theta}_t + \eta_t \gamma) \Big] + \langle \nabla \Phi_t(\Theta_{t-1} + \tilde{\theta}_t), \tilde{\theta}_t - \theta_t \rangle \overset{(a)}{\leq} \Phi(\theta_t - \tilde{\theta}_t) +$$

$$+ \langle \nabla \Phi_t(\Theta_{t-1} + \tilde{\theta}_t), \tilde{\theta}_t - \theta_t \rangle = \max_{y \in \mathcal{X}} \langle y, \theta_t - \tilde{\theta}_t \rangle + \langle \nabla \Phi_t(\Theta_{t-1} + \tilde{\theta}_t), \tilde{\theta}_t - \theta_t \rangle \overset{(b)}{\leq} 2\|\theta_t - \tilde{\theta}_t\|_1, \quad (13)$$

where (a) follows from the sub-additivity of $\Phi(\cdot)$, and in (b) we use that $y_i \in \{0, 1\}, \forall i$ and bounded both terms using triangle inequality. Hence, combining the bounds (12) and (13), we get:

$$\frac{1}{2}\|\theta_t - \tilde{\theta}_t\|_{H_t} \leq \min \Big(\frac{1}{\sqrt{2\pi}} \frac{\|\theta_t - \tilde{\theta}_t\|_1^2}{\eta_t}, 2\|\theta_t - \tilde{\theta}_t\|_1 \Big).$$

Now we choose the learning rate $\eta_t = \beta \sqrt{\sum_{\tau=1}^{t-1} \|\theta_\tau - \tilde{\theta}_\tau\|_1^2}, t \geq 1$ for some constant $0 < \beta \leq \frac{1}{\sqrt{2\pi}}$ that will be specified later. Hence, we have:

$$\frac{1}{2}\|\theta_t - \tilde{\theta}_t\|_{H_t} \leq \min \left(\frac{\|\theta_t - \tilde{\theta}_t\|_1^2}{\sqrt{2\pi}\beta \sqrt{\sum_{\tau=1}^{t-1} \|\theta_\tau - \tilde{\theta}_\tau\|_1^2}}, 2 \frac{\|\theta_t - \tilde{\theta}_t\|_1^2}{\sqrt{2\pi}\beta \sqrt{\|\theta_t - \tilde{\theta}_t\|_1^2}} \right) \overset{(a)}{\leq} \frac{3}{\sqrt{2\pi}\beta} \frac{\|\theta_t - \tilde{\theta}_t\|_1^2}{\sqrt{\sum_{\tau=1}^{t} \|\theta_\tau - \tilde{\theta}_\tau\|_1^2}} \quad (14)$$

where in (a), we used the fact that $\min(a_1/a_2, b_1/b_2) \leq \frac{a_1 + a_2}{b_1 + b_2}$ for any two positive fractions and $\sqrt{x + y} \leq \sqrt{x} + \sqrt{y}$, for any non-negative x and y's. Now that we have a bound for the smoothing-overhead in (10), and the per-step regret bounds (14), we can substitute them in (9) to get:

$$\mathbb{E}_{\gamma}[R_T] \leq \eta_T C \sqrt{2 \log(Ne/C)} + \frac{3}{\sqrt{2\pi}\beta} \sum_{t=1}^{T} \frac{\|\theta_t - \tilde{\theta}_t\|_1^2}{\sqrt{\sum_{\tau=1}^{t} \|\theta_\tau - \tilde{\theta}_\tau\|_1^2}}. \quad (15)$$

The second term above can be upper-bounded as:

$$\sum_{t=1}^{T} \frac{\|\theta_t - \tilde{\theta}_t\|_1^2}{\sqrt{\sum_{\tau=1}^{t} \|\theta_\tau - \tilde{\theta}_\tau\|_1^2}} \leq \sum_{t=1}^{T} \int_{\sum_{\tau=1}^{t-1} \|\theta_\tau - \tilde{\theta}_\tau\|_1^2}^{\sum_{\tau=1}^{t} \|\theta_\tau - \tilde{\theta}_\tau\|_1^2} \frac{dx}{\sqrt{x}} = \int_{0}^{\sum_{\tau=1}^{T} \|\theta_\tau - \tilde{\theta}_\tau\|_1^2} \frac{dx}{\sqrt{x}} = 2\sqrt{\sum_{\tau=1}^{T} \|\theta_\tau - \tilde{\theta}_\tau\|_1^2}.$$

Substituting the above bound into (15) and using the definition of η_T we get the regret upper bound:

$$\mathbb{E}_{\gamma}[R_T] \leq \sqrt{\sum_{\tau=1}^{T} \|\theta_\tau - \tilde{\theta}_\tau\|_1^2} \Big(C\sqrt{2 \ln(Ne/C)}\beta + \frac{6}{\beta\sqrt{2\pi}} \Big).$$

Optimizing over the constant β, and for large cache ($C \geq 11$) we arrive at the result. $\square$

Discussion. Similarly to Theorem 2, the regret bound here is modulated with the quality of predictions: it collapses to zero when predictions are perfect, and maintains $R_T \leq 3.68 \ln \big(\frac{Ne}{C}\big)^{1/4} \sqrt{2CT}$ for arbitrary bad predictions. Interestingly, this worst-case scenario is only inferior by a factor of ~ 2.5 compared to the recent FTPL algorithm in [11] that does not use (and cannot benefit from) predictions. Hence, incorporating predictions (even, of unknown quality) comes without cost in the proposed OFTPL algorithm. Furthermore, for the more common case of predictions within a certain range of error, the regret bounds diminish in proportion to their quality.

Comparing with Theorem 2, OFTPL achieves regret bounds worse by a factor of $\sim 1.9 \ln \big(\frac{Ne}{C}\big)^{1/4}$, which depends on N, albeit in a small order. The regret bounds are also different in nature, for

OFTRL it is ℓ_2 squared whereas for OFTPL, it is the much larger ℓ_1 squared of the prediction error. On the other hand, Algorithm 2 does not involve the expensive projection operation that appears in Algorithm 1, but rather a simple quantile-finding operation (top C files) with a worst-case complexity of $O(N)$. This facilitates greatly the implementation of OFTPL in systems with low computing capacity or in applications that require decisions in near real-time. A notable point about Theorem 3 is that in the special case where the predictor $\tilde{\theta}_t$ suggests a single file (i.e., deterministic), the regret scales with the square root of the *number of mistakes* as opposed to the conventional *number of time slots* (i.e., horizon T) in previous no regret discrete caching works[13][11, 12, 16]. Lastly, it is interesting to note that the bounds of Theorem 3, and 2, provide insights on the appropriate loss function to be optimized by the prediction oracle (squared norms). This is helpful while training and tuning machine learning models on request traces detests.

6 CACHING FILES WITH ARBITRARY SIZES

While the caching problem with equal-sized files has been studied using regret analysis and competitive analysis, to the best of the authors knowledge there are *no results* for the more challenging case of files with different sizes. This section fills this gap by extending the above tools accordingly. In particular, we consider the setting where each file $i \in \mathcal{N}$ has a size of s_i units, $s_i \leq C$. Hence, the set of feasible caching vectors needs to be redefined as:

$$\mathcal{X}_s = \left\{ x \in \{0, 1\}^N \, \middle| \, \sum_{i=1}^{N} s_i x_i \leq C \right\},$$

where the caching decisions are calibrated with the respective file sizes in the capacity constraint. And similarly, the benchmark (designed-in-hindsight) policy is redefined as $x^\star = y^\star \triangleq \operatorname{argmax}_{x \in \mathcal{X}_s} \langle y, \Theta_T \rangle$. We present to solution approaches for this problem, using both OFTRL and OFTPL. These results are of independent interest with applications beyond caching.

6.1 Approximate OFTPL

Similarly to Algorithm 2, the OFTPL algorithm in this case determines the next cache configuration y_t by solving the following integer programming problem at each round t:

$$\mathbb{P}_1: \qquad \max_{y \in \mathcal{X}_s} \langle \Theta_{t-1} + \tilde{\theta}_t + \eta_t \gamma, y \rangle,$$

which is a Knapsack instance with profit vector $p = \Theta_{t-1} + \tilde{\theta}_t + \eta_t \gamma$; size vector $s = (s_i, \forall i \in \mathcal{N})$; and capacity C. Since the Knapsack problem is NP-Hard [63], we cannot solve $\mathbb{P}_1$ efficiently (fast and accurately) at each slot, and hence it is not practical (or, even possible) to use the approach of Sec. 5. Instead, we resort here to an approximation scheme for solving $\mathbb{P}_1$ and, importantly, do so in a way that these approximately-solved instances do not accumulate an unbounded regret w.r.t. $y^\star$. This requires a tailored approximation analysis and to define a new regret metric.

In detail, we leverage Dantzig's approach for tackling packing problems [29], to obtain an *almost*-integral solution from the respective integrality-relaxed problem; and then recover, via a *point-wise randomized rounding*, a fully-integral solution which, as we prove, keeps the long-term $1/2$-approximate regret bounded. First, recall that the α-approximate regret is defined as [64]:

$$R_T^{(\alpha)} \triangleq \alpha \langle \Theta_T, x^\star \rangle - \sum_{t=1}^{T} \langle \theta_t, x_t \rangle, \qquad (16)$$

---

[13]The same property of depending on prediction mistakes rather than T holds for Theorem 2 but the regret scales as square root of double the number of mistakes, due to the use of ℓ_2 norm.

Algorithm 3: `OFTPL-UneqCache`

---

1 **Input**: $\eta_1 = 0$, $y_1 \in \mathcal{X}$, $s = (s_i, i \in \mathcal{N})$
2 **Output**: $\{y_t \in \mathcal{X}_s\}_T$ // *Feasible discrete caching vector at each slot*
3 $\gamma \overset{iid}{\sim} \mathcal{N}(0, \mathbf{1}_{N \times 1})$ // *Sample a perturbation vector*
4 **for** $t = 2, 3, \ldots$ **do**
5 $\quad$ $\tilde{\theta}_t \leftarrow$ prediction   //*Obtain request prediction for slot t*
6 $\quad$ $\eta_t = \frac{1.3}{\sqrt{C}} \left(\frac{1}{\ln(Ne/C)} \right)^{1/4} \sqrt{\sum_{\tau=1}^{t-1} \|\theta_\tau - \tilde{\theta}_\tau\|_1^2}$ // *Update the perturbation parameter*
7 $\quad$ $p \leftarrow \Theta_{t-1} + \tilde{\theta}_t + \eta_t \gamma$ // *Update the profit vector*
8 $\quad$ $(\hat{y}_t, k) \leftarrow$ Dantz(C, p, s) // *Compute the "almost integral" cache vector*
9 $\quad$ $y_t \leftarrow$ Rand$(\hat{y}_t, k)$ // *Perform Randomized Rounding*
10 $\quad$ $\Theta_t = \Theta_{t-1} + \theta_t$ // *Receive t-slot request and update total grad*
$\quad$ **end**

---

for a positive constant α. This generalized regret metric allows to use a parameterized benchmark, in line with prior works, e.g., see [16] and references therein. Now, it is important to see that while the Knapsack problem admits an FPTAS [65], due to the online nature of our caching problem, not all α-approximation schemes for the offline OFTPL problem provide an α-approximate regret guarantee. In light of this, we employ the stronger notion of *point-wise α-approximation scheme*, which yields an α-regret guarantee for the online learning problem [64]. We restate the definition:

DEFINITION 1 (α-POINT-WISE APPROXIMATION). *A randomized α-point-wise approximation algorithm $\mathcal{A}$ for a fractional solution $\hat{y} = (\hat{y}_i, i \in \mathcal{N})$ of a maximizing LP with non-negative coefficients, is one that returns an integral solution $y = (y_i, i \in \mathcal{N})$ such that $\mathbb{E}[y_i] \geq \alpha \hat{y}_i, \forall i \in \mathcal{N}$ and some $\alpha > 0$; where the expectation is taken over possible random choices made by algorithm $\mathcal{A}$.*

In our case, we set $\alpha = 1/2$ and propose an ($1/2$)-point-wise approximation algorithm for $\mathbb{P}_1$.

Our starting point is Dantzig's approach which operates on the integrality-relaxed version of $\mathbb{P}_1$. In particular, the integrality-relaxed LP for the Knapsack problem with profit vector p, weight vector s, and capacity C, through the following steps:

Dantz(C, p, s):

---

(1) Index files in decreasing profit-to-size ratios, *i.e.*, $(p_1/s_1) \geq (p_2/s_2) \geq \ldots \geq (p_N/s_N)$.
(2) Set $k = \min \left\{ j \,\middle|\, \sum_{i=1}^{j} s_i > C \right\}$ and $\widetilde{C} = C - \sum_{i=1}^{k-1} s_i$.
(3) Assign the continuous variables $\hat{y}_i \in [0, 1], i \in \mathcal{N}$ as follows:

$$\hat{y}_i = \begin{cases} 1, & \text{if } i \in [k-1] \\ \frac{\widetilde{C}}{s_k}, & \text{if } i = k \\ 0, & \text{otherwise.} \end{cases}$$

To streamline presentation, we denote with Dantz(C, p, s) the operation of steps (1)-(3) above on the Knapsack instance (C, p, s), which return the solution $\hat{y}$ and the value of parameter k. An interesting property of returned solution $\hat{y}$, which we exploit in our randomized approximation scheme, is that at most one component of the vector $\hat{y}$ is non-integral.

The detailed steps of the proposed OFTPL scheme are presented in Algorithm 3. At each slot we obtain a new (probabilistic) prediction for the next requested file $\tilde{\theta}_t$ (line 5) and update the perturbation parameter η_t (line 6). Then we calculate the new profits $p_i = \Theta_{t-1} + \tilde{\theta}_t + \eta_t \gamma, i \in \mathcal{N}$, and solve the relaxed Knaspack by invoking Dantz(C, p, s) to obtain the almost-integral $\hat{y}_t$ and parameter k (line 7). This vector has $k-1$ components equal to 1, one additional non-negative component, and $N - k$ components equal to 0. This solution is then rounded through the randomization scheme:

Rand($\hat{y}_t, k$):
(1) Set $S = 1, 2, \ldots, k - 1 \triangleq [k-1]$ with probability ½; Set $S = \{s\}$ with probability ½
(2) Set $y_{ti} = 0, \forall i \in \mathcal{N}$; and update to $y_{ti} = 1$ for each $i \in S$.

The Rand operation is invoked and creates integral caching vector y_t which satisfies the capacity constraint (line 9). Finally, we observe the new gradient, update the aggregate gradient vector and repeat the process (line 10). The following theorem, proved in Appendix. A.4, characterizes the guarantees of Algorithm 3.

THEOREM 4. *Algorithm 3 ensures, for any time horizon T, the expected regret bound:*

$$\mathbb{E}\left[R_T^{(1/2)}\right] \le 1.84\sqrt{C}\left(\ln \frac{Ne}{C}\right)^{1/4}\sqrt{\sum_{t=1}^{T}\|\theta_t - \tilde{\theta}_t\|_1^2}$$

Discussion. The bounds of Theorem 3 posses the desirable property of being modulated with the prediction errors, and in fact are improved by a factor of half compared to the equal-sizes bound. However, we remind the reader that the regret metric in this section is defined w.r.t. a weaker benchmark, i.e., a benchmark that achieves ½ the utility of the best-in-hindsight utility $\langle \Theta_T, y^\star \rangle$. Relying to such approximations is an inevitable concession for NP-hard problems – especially when having to solve them repeatedly. Interestingly, the computational complexity of Algorithm 3 is comparable to that of Algorithm 2, which is quite surprising since we are able to handle integral caching decisions with arbitrary-sized files. To the best of the authors knowledge, this work is the first to propose an OCO-based solution for this variant of the online caching problem. We proceed next to remove the effect of the library size on the regret bound.

6.2 Approximate OFTRL

We introduce next an OFTRL algorithm that can handle arbitrary file sizes. To that end, we use the OFTRL update on the integrality-relaxed version of the problem, then modify the obtained vector so as to be almost integral, and finally employ the same randomized rounding technique as above to correct for feasibility. The new intermediate step (that yields the almost-integral vector) is necessary since, unlike in Sec. 6.1, we cannot apply the (½)-sampling technique directly on $\{\hat{x}_t\}_t$ because these vectors do not have this required almost-integral form. Thus, we leverage the DepRound subroutine from [28], which is known to achieve the useful property re-stated below.

LEMMA 2. *For $a \in [0, 1]^N$, the DepRound scheme in Algorithm 8 returns a vector b such that*

- *All elements of b are integral except one: $b_i \in \{0, 1\} \; \forall i \in \mathcal{N}/j, \; b_j \in [0, 1]$.*
- *b_i is an unbiased estimator of a_i: $\mathbb{E}[b_i] = a_i, \forall i \in \mathcal{N}$.*
- *b respects the linear constraints satisfied by a: $\Pr[\sum_{i \in \mathcal{N}} s_i a_i = \sum_{i \in \mathcal{N}} s_i b_i] = 1$.*

With this almost-integral solution at hand, we re-use the sampling technique Rand, to achieve an (½)-Regret guarantee w.r.t. the same benchmark as in Sec. 6.1. The steps are presented in Algorithm 4. The diligent reader will observe that essentially we merge steps from Algorithm 1 and Algorithms 3. The detailed steps of the DepRound subroutine are presented in the Appendix. The next theorem, proved in the Appendix A.5, characterizes the algorithm's performance.

THEOREM 5. *Algorithm 4 ensures, for any time horizon T, the expected regret bound:*

$$\mathbb{E}\left[R_T^{(1/2)}\right] \le \sqrt{C}\sqrt{\sum_{t=1}^{T}\|\theta_t - \tilde{\theta}_t\|_2^2}$$

Algorithm 4: `OFTRL-UneqCache`

---

1   **Input**: $\sigma = 1/\sqrt{C}$, $\delta_1 = \|\theta_1 - \tilde{\theta}_1\|_2^2$, $\sigma_1 = \sigma\sqrt{\delta_1}$, $x_1 \in \mathcal{X}_s$

2   **Output**: $\{y_t \in \mathcal{X}_s\}_T$ *// Feasible discrete caching vector at each slot*

3   **for** $t = 2, \ldots$ **do**

4      $\tilde{\theta}_t \longleftarrow$ prediction   *//Obtain gradient prediction for slot t*

5      $\hat{x}_t = \arg\max_{x \in \text{conv}(\mathcal{X}_s)} \left\{ -r_{1:t-1}(x) + \langle x, \Theta_{t-1} + \tilde{\theta}_t \rangle \right\}$

6      $\bar{x}_t \longleftarrow$ DepRound$(\hat{x}_t)$ *// Compute the "almost integral" cache vector*

7      $x_t \longleftarrow$ Rand$(\bar{x}_t, k)$ *// Perform Randomized Rounding*

8      $\Theta_t = \Theta_{t-1} + \theta_t$ *// Receive t-slot request and update total grad*

9      $\sigma_t = \sigma \left(\sqrt{\delta_{1:t}} - \sqrt{\delta_{1:t-1}} \right)$ *// Update the regularization parameter*

   end

Discussion. Compared to the approach we used to extend OFTPL to unequal-sized files, an additional rounding technique (DepRound) was necessary to extend OFTRL. The complexity of this sub-routine is linear in the library size. Thus, despite preserving the order-level complexity of Algorithm 4, handling such files increases the overhead to get the almost-integral caching vectors $\{\bar{x}_t\}_t$. The important point is that we recover the dimension-free bounds for the regret, which are better by a constant factor of $1.84 \left(\ln \frac{Ne}{C} \right)^{1/4}$ compared to `OFTPL-UneqCache`, at the expense of performing a (potentially challenging) projection step. Hence, one can select the most suitable method depending on the requirements of the application (size of library, slot length, etc.) and the computing resources when executing the algorithm. Lastly, in Algorithm 4, and all introduced algorithms for the single cache case, the request vector θ can be extended to include (time-varying and unknown) weights that depend on users or network properties.

7   BIPARTITE CACHING THROUGH OPTIMISM (`OFTRL-BIPCACHE`)

Finally, we extend our study to caching networks where a set of edge caches $\mathcal{J} = \{1, 2, \ldots, J\}$ serves a set of users $\mathcal{I} = \{1, 2, \ldots, I\}$ requesting files from the library $\mathcal{N}$. The connectivity between $\mathcal{I}$ and $\mathcal{J}$ is modeled with parameters $d = \left(d_{ij} \in \{0, 1\} : i \in \mathcal{I}, j \in \mathcal{J} \right)$, where $d_{ij} = 1$ if cache j can be reached from user location i. Each user can be (potentially) served by any connected cache. This is a widely-studied non-capacitated bipartite model [2, 8].

We introduce the new caching variables k and routing variables u. Namely, $k_{nj}^t \in \{0, 1\}$ decides whether file $n \in \mathcal{N}$ is stored at cache $j \in \mathcal{J}$ at the beginning of slot t, and the t-slot caching vector $k_t = (k_{nj}^t : n \in \mathcal{N}, j \in \mathcal{J})$ belongs to $\mathcal{K} = \left\{ k \in \{0, 1\}^{N \cdot J} \mid \sum_{n \in \mathcal{N}} k_{nj} \leq C_j, \ j \in \mathcal{J} \right\}$, where C_j is the capacity of cache $j \in \mathcal{J}$. We use the routing variable $u_{nij}^t \in \{0, 1\}$ to decide the delivery of file n to user i from cache j, and define the t-slot routing vector $u_t = (u_{nij}^t \in [0, 1] : n \in \mathcal{N}, i \in \mathcal{I}, j \in \mathcal{J})$ that is selected from the set: $\mathcal{U} = \left\{ u \in \{0, 1\}^{N \cdot J \cdot I} \mid \sum_{j \in \mathcal{J}} u_{nij} \leq 1, \ n \in \mathcal{N}, i \in \mathcal{I} \right\}$. Note also that unserved requests, i.e., when the summation is strictly smaller than 1, are satisfied by the root cache. This option, however, yields zero benefit for the users (no cache-hit gains); see also [2]. The request vector θ_t is redefined to reflect a request's source and destination: $\theta_t = (\theta_{ni}^t \in \{0, 1\} : n \in \mathcal{N}, i \in \mathcal{I})$, and is drawn from the set: $Q = \left\{ \theta \in \{0, 1\}^{N \cdot I} \mid \sum_{n \in \mathcal{N}} \sum_{i \in \mathcal{I}} \theta_{ni} = 1 \right\}$.

We can now introduce the t-slot utility function:

$$f_t(x_t) = \sum_{n \in \mathcal{N}} \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{J}} \theta_{ni}^t k_{nij}^t .$$

Algorithm 5: `OFTRL-BipCache`

---

1 **Input**: $\sigma = 1/\sqrt{1+JC}$, $\delta_1 = \|\theta_1 - \tilde{\theta}_1\|_2^2$, $\sigma_1 = \sigma\sqrt{\delta_1}$, $x_1 \in \mathcal{X}_b$

2 **Output**: $\{x_t = (k_t, u_t) \in \mathcal{X}_b\}_T$ *// Feasible discrete caching/routing vector at each slot*

3 **for** $t = 2, 3, \ldots$ **do**

4      $\tilde{\theta}_t \leftarrow$ prediction   *//Obtain request prediction for slot t*

5      $\hat{x}_t = \arg\max_{x \in \mathrm{conv}(\mathcal{X}_b)} \left\{ -r_{1:t-1}(x) + \langle x, \Theta_{t-1} + \tilde{\theta}_t \rangle \right\}$ *// Update the continuous policy vector*

6      $(k_j^t) \longleftarrow MadowSample(\hat{k}_t), \forall j \in \mathcal{J}$ *// Obtain the discrete cache vector for each cache independently*

7      $\Theta_t = \Theta_{t-1} + \theta_t$ *// Receive t-slot request and update total grad*

8   Set $u_{nij'}^t \leftarrow 1$ for a randomly selected $j' \in \mathcal{J}^{ni}$ *//Update the discrete routing vector*

9      $\sigma_t = \sigma\left(\sqrt{\delta_{1:t}} - \sqrt{\delta_{1:t-1}}\right)$ *// Update the regularization parameter*

    end

---

where we abuse notation and redefine $x_t \triangleq (k_t, u_t)$. Therefore, the utility-maximizing caching-routing policy at each slot t is found by solving the following problem:

$$\mathbb{P}_2 : \max_x \sum_{t=1}^{T} f_t(x) \quad \text{s.t.} \quad u \in \mathcal{U}, \quad k \in \mathcal{K}; \quad u_{nij} \leq k_{nj} d_{ij}, \, i \in \mathcal{I}, j \in \mathcal{J}, n \in \mathcal{N},$$

and we define the feasible caching/routing set as $\mathcal{X}_b \triangleq \left\{ \{\mathcal{K} \times \mathcal{U}\} \cap \{u_{nij} \leq k_{nj} d_{ij}\} \right\}$. $\mathbb{P}_2$ is known to be NP-Hard via a reduction to the set cover problem [2, Sec. 3], [16, Sec. 4.1]. Hence, we will be using below also its integrality-relaxed version $\mathbb{P}_2'$ with continuous variables $\hat{x}_t \triangleq (\hat{k}_t, \hat{u}_t)$.

Our strategy for tackling $\mathbb{P}_2$ is to use OFTRL on the convex hull of $\mathcal{X}_b$ (essentially learning w.r.t. $\mathbb{P}_2'$) to optimize $\hat{x}_t$, and then obtain discrete caching vectors with Madow's sampling applied to each cache separately. As last step we select a proper routing solution for the received request θ_t. Namely, upon receiving a request for file n from user i, the corresponding routing variable is set to 1 if *any* cache connected to i stores file n. Thus, we define the auxiliary set $\mathcal{J}^{ni} = \left\{ j \in \mathcal{J} \,\middle|\, y_{nj} d_{ij} = 1 \right\}$, and assign $u_{nij'}^t = 1$ for the (n, i) pair and some $j' \in \mathcal{J}^{ni}$. It is important to stress that in such uncapacitated models, the routing plan is directly determined once a caching vector is fixed[14]. The detailed steps of the proposed OFTRL schemed are presented in Algorithm 5, where we reuse the regularization scheme from Sec. 4 with the difference that it operates now on the newly defined variables and request vectors. The performance of the algorithm is characterized with the next theorem, the proof of which can be found in Appendix. A.6

THEOREM 6. *Algorithm 5 ensures, for any horizon T, the expected $(1 - 1/e)-$Regret bound:*

$$\mathbb{E}\left[R_T^{(1-1/e)}\right] \leq 1.3\sqrt{1+JC}\sqrt{\sum_{t=1}^{T} \|\theta_t - \tilde{\theta}_t\|_2^2}$$

Discussion. Similar to the single cache, Theorem 6 improves the regret by removing the effect of library size N, as opposed to the recently-proposed bipartite OFTPL algorithm for equal-sized files in [16][15]. This improvement comes at the expense of a projection operation in the OFTRL step. Additionally, Algorithm 5 manages to reduce further the constant terms when it has access to high quality predictions for the next-slot requests.

---

[14]In other words, the routing variables are auxiliary in the Femtocaching model, and indeed in [2] these variables where omitted, while they appear with different name in subsequent works, e.g., as virtual caching variables in [16].

[15]We note that the bound in [16] contains additionally the number of users since they consider a different request model of one request per user per time slot. Our work can be readily extended in that direction, as explained above.

8 EXPERT-BASED OPTIMISTIC CACHING

Changing tack in this section, we explore a different approach for optimism that is based on the classical experts framework. Specifically, we consider a model with two experts: a pessimistic (or robust) learner and an optimistic learner. The pessimist expert *proposes* caching decisions based on the OGA policy [14] that does not use predictions, and provides adversarial regret guarantees. The optimistic expert proposes a caching policy that is optimized solely w.r.t. the predicted request, i.e., as if the predictions are fully reliable. Finally, a meta-learner receives the proposals from the two experts and gradually discerns which of them should be trusted. The expert-based approach to optimistic learning has been previously proposed for *continuous* caching in [15, 17]. We expand it here to handle discrete decisions and demonstrate it using the single cache scenario.

Formally, the pessimistic expert (p) proposes caching $\{\hat{z}_t^{(p)}\}_t$ according to *adaptive* OGA:

$$\hat{z}_t^{(p)} = \mathcal{P}_{\text{conv}(\mathcal{X})} \left\{ z_{t-1}^{(p)} + \frac{1}{\sqrt{t}} \theta_t \right\}, \tag{17}$$

where $\mathcal{P}_{\text{conv}(\mathcal{X})}$ is the Euclidean projection onto the convex hull of $\mathcal{X}$. We denote the regret of this expert by $R_T^{(p)} = \langle \Theta_T, \hat{z}^\star \rangle - \sum_{t=1}^T \langle \theta_t, \hat{z}_t^{(p)} \rangle$, where $z^\star = \text{argmax}_{z \in \text{conv}(\mathcal{X})} \langle \Theta_T, z \rangle$. On the other hand, the optimistic expert (o) solves the following LP $z^{(o)} = \arg \max_{z \in \mathcal{X}} \langle \tilde{\theta}_t, z \rangle$, and we denote its regret with $R_T^{(o)} = \langle \Theta_T, z^\star \rangle - \sum_{t=1}^T \langle \theta_t, z_t^{(o)} \rangle$.

Unlike the previous sections where predictions were used to modify the perturbation and regularization parameters, here they are treated independently through the optimistic expert. The challenge is then to learn which of the two experts' proposals to follow. To that end, a meta-learner combines the proposals through a set of learned weights. Namely, the meta-learner's decision variable is $w \triangleq (w^{(p)}, w^{(o)}) \in \Delta_2$, where $\Delta_2 = \{ w \in [0,1]^2 \big| \|w\|_1 = 1 \}$, and is used to create a convex combination of the provided caching proposals, i.e., $\hat{z}_t = w_t^{(p)} \hat{z}_t^{(p)} + w_t^{(o)} z_t^{(o)}$. Clearly, by its definition, it holds that $\hat{z}_t \in \text{conv}(\mathcal{X})$. The weights are updated with adaptive OGA:

$$\hat{w}_t = \mathcal{P}_{\text{conv}(\Delta_2)} \left\{ w_{t-1} + \frac{1}{\sqrt{t}} l_t \right\}, \tag{18}$$

where $l_t \triangleq \left(\langle \theta_t, \hat{z}_t^{(p)} \rangle, \langle \theta_t, z_t^{(o)} \rangle \right)$ is the experts' utility vector at slot t. We then have the following result for the regret of the actual mixed action [17, Thm. 3]:

$$\hat{R}_T\{\hat{z}\}_T = \langle \Theta_T, \hat{z}^\star \rangle - \sum_{t=1}^T \langle \theta_t, \hat{z}_t \rangle \leq R_T^{(w)} + \min \left\{ R_T^{(p)}, R_T^{(o)} \right\} \leq 2\sqrt{2T} + \min \left\{ R_T^{(p)}, R_T^{(o)} \right\} \tag{19}$$

Finally, similar to what we have shown in the OFTRL section, it is possible to use Madow's sampling to recover integral cache states $z_t \in \mathcal{X}$ with the associated bound

$$\mathbb{E}\left[R_T\{z_t\} \right] \leq R_T^{(w)} + \min \left\{ R_T^{(p)}, R_T^{(o)} \right\}. \tag{20}$$

The steps of this scheme are summarized in Algorithm 6.

Discussion. The performance advantage of the bound in (20) is that it can be strictly negative, depending on the optimistic expert's regret. For example, in case of perfect predictions and non-fixed cost functions, the min term evaluates to $-\epsilon T$ for some $\epsilon > 0$, making the meta-regret negative for large enough T. In all cases, the meta-regret is upper bounded by $O(\sqrt{T})$ due to the existence of the robust expert's regret in the min term, hence we maintain the order-optimal regret for worst-case scenarios with this approach as well. From a computational load perspective, the most challenging step is the projection involved in the calculation of the OGD-based policy (pessimistic expert). However, one can leverage the tailored fast projection proposed in [14] for that operation.

Algorithm 6: Experts-Cache

---

1 **Input**: $z_1 \in \mathcal{X}$

2 **Output**: $\{z_t \in \mathcal{X}\}_T$ // *Feasible caching vector at each slot*

3 **for** $t = 2, 3, \ldots$ **do**

4      $\hat{z}_t^{(p)} = \mathcal{P}_{\text{conv}(\mathcal{X})} \left\{ z_{t-1}^{(p)} + \frac{1}{\sqrt{t}} \theta_t \right\}$ // *Pessimistic experts makes proposal*

5      $z^{(o)} = \arg\max_{z \in \mathcal{X}} \langle \tilde{\theta}_t, z \rangle$ // *Optimistic experts makes proposal*

6      $\hat{z}_t = w_t^{(p)} \hat{z}_t^{(p)} + w_t^{(o)} z_t^{(o)}$ // *Meta-learner combines proposals*

7      $z_t \leftarrow MadowSample(\hat{z}_t)$ // *Obtain the discrete cache vector using Algorithm 7*

8      $\Theta_t = \Theta_{t-1} + \theta_t$ // *Receive the request for slot t and update total grad*

9      $\hat{w}_t = \mathcal{P}_{\text{conv}(\Delta_2)} \left\{ w_{t-1} + \frac{1}{\sqrt{t}} l_t \right\}$ // *Meta-learner observes losses l_t & updates weights*

    end

It is also important to stress that this framework allows to combine more than one experts, in order to either to e.g., include more than one predictors, see discussion also in [15].

We note that since experts-based optimism is a meta-algorithm whose regret is characterized by that of the experts (i.e., learning algorithms), it can be applied to the other setups of unequal sizes and bipartite caching. The (possibly α) meta regret will then be related to that of the (possibly α) regret of the optimistic and pessimistic experts. Finally, it is worth noting that the idea of using the experts model for combining multiple caching policies has been previously proposed in [66], and evaluated in several cases, e.g., see [67] and reference therein, which however do not consider predictors nor provide any theoretical analysis (or, bounds) for the performance of this approach.

9 EXPERIMENTS

We compare the performance of our algorithms with carefully-selected competitors: the FTRL policy which generalizes the OGD from [14] and the FTPL method from [11]. We note that these competitors already showed superior performance to the classical methods of LRU and LFU in their experiments. The request traces are created using the MovieLens dataset [68] which contains time-stamped movie ratings. We assume a request is initiated to a CDN in the same chronological order as their ratings' timestamps. We consider movies with at least 8 ratings, leading to a library of $N = 10379$ and we set capacity $C = 150$. Each prediction is assumed correct with probability ρ. Specifically, we generate a one-hot $\tilde{\theta}_t$ that has 1 at the file to be requested with probability ρ, or at any other random file with probability $1 - \rho$. We also experiment with *probabilistic* predictions where the vector components represent the probabilities of files being requested (details in Appendix A.7.1). For the experiments with unequal-sized files, we generate the sizes uniformly $s_i \sim U[1, 10]$ and set $C = 500$. For the bipartite network, we use the 100k variation of the MovieLens dataset and consider files with at least 10 ratings, leading to $N = 1152$. The network consists of 3 caches ($C = 150$) and 4 user locations, the first two connected to caches 1 & 2, and the rest to caches 2 & 3.

Fig. 2 shows the average regret (hit-rate gap to the optimal) growth with time for FTPL [11], FTRL [14], and their proposed optimistic counterparts. We experiment with $\rho = 0$, and $\rho = 0.5$. If, e.g., the request predictions were based on recommendations, these reflect the cases where the users do not follow the recommendations ($\rho = 0$), or actually request the recommended movie/file with probability 50%, ($\rho = 0.5$). We observe that optimism accelerates and improves learning the best files to cache, reaching an average improvement of 63.8% (for OFTRL) and 68.8% (for OFTPL) compared to their "vanilla" counterparts (no predictions). Moreover, the performance degradation due to inaccurate predictions is almost negligible: < 1.9% for OFTRL and < 3% for OFTPL. We also plot the 0.95-confidence interval of R_T in Figures 2c and 2d, where we note the more condensed
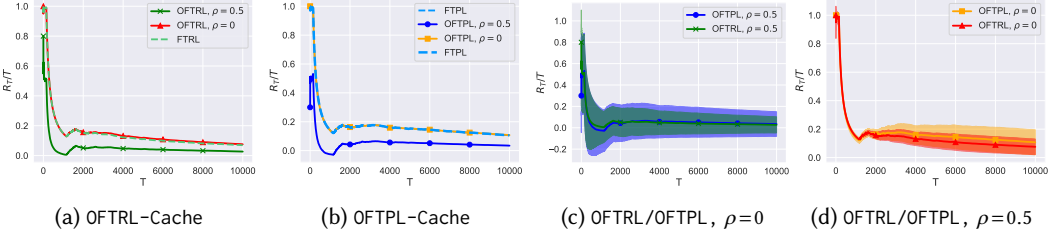
Fig. 2. Comparison of R_T/T for **equal-sized** files in a **single cache** and different policies: (a) `OFTRL-Cache` vs. FTRL/OGD; (b) `OFTPL-Cache` vs. FTPL [11]; (c) `OFTRL-Cache` vs. OFTPL-Cache for good predictions and in (d) for worst-case predictions. In (c), (d) we plot the 0.95-confidence interval (10 runs).
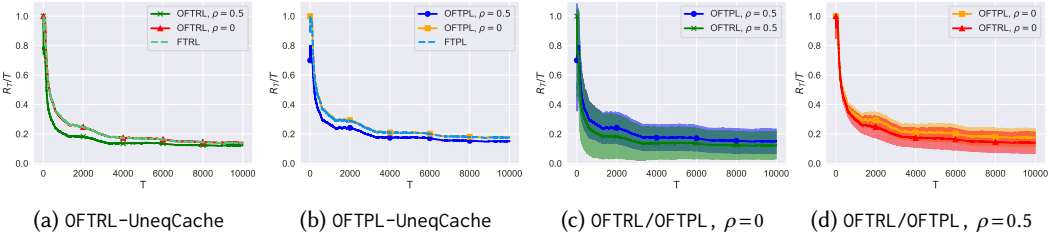

Fig. 3. Comparing R_T/T for **unequal-sized** files, **single cache**. (a) `OFTRL-UneqCache` vs. FTRL/OGD; (b) `OFTPL-UneqCache` vs. FTPL [11]; (c)-(d) `OFTRL-UneqCache` vs. OFTPL-UneqCache for good (bad) predictions.

distribution for OFTRL: 34.6% and 37.8% tighter at $t = 10k$ when $\rho = 0.5, \rho = 0$, respectively. This is because the distribution ($\{\hat{x}_t\}_T$ iterates in OFTRL) become more concentrated with time; an argument that is not directly applicable to OFTPL, where the randomness is due to solving a *perturbed* linear program. In Fig. 3 we evaluate the algorithms for the *unequal* sizes case and plot the ½-regret. We observe the same pattern of negligible performance degradation when $\rho = 0$, while $\rho = 0.5$ enables an improvement of 22.7% (for OFTRL) and 16.5% (for OFTPL). We kindly refer the reader to the appendix for additional experiments for the experts-caching algorithm, the bipartite caching problem, and with probabilistic predictions of varying qualities.

10 CONCLUSIONS AND FUTURE WORK

In this paper, we presented several provably optimal algorithms that exploit predictions of unknown quality to improve the regret bounds for important variants of the discrete caching problem, while maintaining worst-case guarantees. The tackled problems are general (e.g., the Knapsack problem) and extend beyond caching; and hence the corresponding proposed optimistic algorithms can be applied to other similar problems. Our approach was based on the unified view of FTRL and FTPL algorithms as smoothing operations, where we proposed to make such smoothing adaptive to the predictions' accuracy. This allowed us to obtain a regret that interpolates between 0 and $O(\sqrt{T})$.

This work also paves the road for several promising extensions. Given that eviction-only policies such as LFU or LRU have provably linear worst-case regret [11], we studied policies that can dynamically prefetch files. Thus, balancing the cache hits with prefetching costs remains to be tackled. Moreover, we note that *static regret* algorithms, like ours, can be used as a subroutine in algorithms with stronger benchmarks, such as the Φ-regret [69] and the minimum regret over all Finite-State-Predictors [70], and extending the study towards such more-refined benchmarks is certainly interesting. Finally, considering unequal routing utility (e.g., link-capacitated model [71]) and unequal-sized files for the bipartite network model remains an open question [72].

REFERENCES

[1] T. Bektas, O. Oguz, and O. I., "Designing Cost-effective Content Distribution Networks," *Computers & Operations Research*, vol. 34, pp. 2436–2449, 2007.

[2] K. Shanmugam, N. Golrezaei, A. G. Dimakis, A. F. Molisch, and G. Caire, "Femtocaching: Wireless Content Delivery Through Distributed Caching Helpers," *IEEE Trans. Inform. Theory*, vol. 59, no. 12, pp. 8402–8413, 2013.

[3] D. Chatzopoulos, C. Bermejo, Z. Huang, and P. Hui, "Mobile Augmented Reality Survey: From Where We Are to Where We Go," *IEEE Access*, vol. 5, pp. 6917–950, 2017.

[4] G. S. Paschos, G. Iosifidis, M. Tao, D. Towsley, and G. Caire, "The Role of Caching in Future Communication Systems and Networks," *IEEE J. Select. Areas Commun.*, vol. 36, no. 6, pp. 1111–1125, 2018.

[5] G. Gracioli, A. Alhammad, R. Mancuso, A. A. Frohlich, R. Pellizzoni, "A Survey on Cache Management Mechanisms for Real-Time Embedded Systems," *ACM Comput. Surv.*, vol. 48, no. 2, pp. 32:1–32:36, 2015.

[6] P. R. Jelenković and X. Kang, "Characterizing the Miss Sequence of the LRU Cache," *SIGMETRICS Perform. Eval. Rev.*, vol. 36, no. 2, pp. 119–121, 2008.

[7] D. Lee, J. Choi, J.-H. Kim, S. H. Noh, S. L. Min, Y. Cho, and C. S. Kim, "On the Existence of a Spectrum of Policies That Subsumes the Least Recently Used (LRU) and Least Frequently Used (LFU) Policies," *SIGMETRICS Perform. Eval. Rev.*, vol. 27, no. 1, pp. 134–143, 1999.

[8] G. Paschos, G. Iosifidis, and G. Caire, "Cache Optimization Models and Algorithms," *FnT in Communications and Information Theory*, vol. 16, no. 3–4, pp. 156–345, 2020.

[9] F. Guillemin, T. Houdoin, and S. Moteau, "Volatility of youtube content in orange networks and consequences," in *2013 IEEE International Conference on Communications (ICC)*, 2013, pp. 2381–2385.

[10] E. Hazan, "Introduction to Online Convex Optimization," 2019. [Online]. Available: https://arxiv.org/abs/1909.05207

[11] R. Bhattacharjee, S. Banerjee, and A. Sinha, "Fundamental Limits on the Regret of Online Network-Caching," *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 4, no. 2, 2020.

[12] T. Si Salem, G. Neglia, and S. Ioannidis, "No-regret caching via online mirror descent," 2021. [Online]. Available: https://arxiv.org/abs/2101.12588

[13] ——, "No-Regret Caching via Online Mirror Descent," in *Proc. of ICC*, 2021.

[14] G. S. Paschos, A. Destounis, and G. Iosifidis, "Online Convex Optimization for Caching Networks," *IEEE/ACM Trans. Networking*, vol. 28, no. 2, pp. 625–638, 2020.

[15] N. Mhaisen, G. Iosifidis, and D. Leith, "Online caching with no regret: Optimistic learning via recommendations," 2022. [Online]. Available: https://arxiv.org/abs/2204.09345

[16] D. Paria and A. Sinha, "Leadcache: Regret-optimal caching in networks," in *Proc. of NeurIPS*, 2021.

[17] N. Mhaisen, G. Iosifidis, and D. Leith, "Online Caching with Optimistic Learning," in *Proc. of IFIP Networking*, 2022.

[18] F. Orabona, "A Modern Introduction to Online Learning," 2019. [Online]. Available: https://arxiv.org/abs/1912.13213

[19] M. A. Maddah-Ali, and U. Niesen, "Fundamental Limits of Caching," *IEEE Trans. Inf. Theory*, vol. 60, no. 5, pp. 2856–2867, 2014.

[20] C. A. Gomez-Uribe and N. Hunt, "The Netflix Recommender System: Algorithms, Business Value, and Innovation," *ACM Trans. Manage. Inf. Syst.*, vol. 6, no. 4, 2016.

[21] X. Amatriain, "Building Industrial-Scale Real-World Recommender Systems," in *Proc. of RecSys*, 2012.

[22] Y. Li, T. Si Salem, G. Neglia, and S. Ioannidis, "Online caching networks with adversarial guarantees," *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 5, no. 3, 2021.

[23] A. Rakhlin and K. Sridharan, "Optimization, Learning, and Games with Predictable Sequences," in *Proc. of NeurIPS*, 2013.

[24] M. Mohri and S. Yang, "Accelerating Online Convex Optimization via Adaptive Prediction," in *Proc. of AISTATS*, 2016.

[25] W. Wang, and C. Lu, "Projection onto the Capped Simplex," *arXiv preprint arXiv:1503.01002*, 2015.

[26] T. Lykouris and S. Vassilvtiskii, "Competitive Caching with Machine Learned Advice," in *Proc. of ICML*, 2018.

[27] D. Rohatgi, "Near-optimal bounds for online caching with machine learned advice," in *Proc. of ACM-SIAM SODA*, 2020, pp. 1834–1845.

[28] J. Byrka, T. Pensyl, B. Rybicki, A. Srinivasan, and K. Trinh, "An improved approximation for k-median and positive correlation in budgeted optimization," *ACM Trans. Algorithms*, vol. 13, no. 2, 2017.

[29] G. B. Dantzig, "Discrete-variable extremum problems," *Operations Research*, vol. 5, no. 2, pp. 266–277, 1957.

[30] E. Leonardi, and G. Neglia, "Implicit Coordination of Caches in Small-cell Networks Under Unknown Popularity Profiles," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 6, pp. 1276–1285, 2018.

[31] A. Giovanidis, and A. Avranas, "Spatial Multi-LRU: Distributed Caching for Wireless Networks with Coverage Overlaps," *arXiv:1612.04363*, 2016.

[32] S. Traverso, M. Ahmed, M. Garetto, P. Giaccone, E. Leonardi, and S. Niccolini, "Temporal Locality in Today's Content Caching: Why It Matters and How to Model It," *SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 5, pp. 5–12, 2013.

[33] F. Olmos, B. Kauffmann, A. Simonian, and Y. Carlinet, "Catalog dynamics: Impact of content publishing and perishing on the performance of a LRU cache," in *Proc. of ITC*, 2014.

[34] M. Leconte, G. Paschos, L. Gkatzikis, M. Draief, S. Vassilaras, and S. Chouvardas, "Placing Dynamic Content in Caches with Small Population," in *Proc. of IEEE INFOCOM*, 2016.

[35] A. Sadeghi, F. Sheikholeslami, and G. B. Giannakis, "Optimal and Scalable Caching for 5G Using Reinforcement Learning of Space-Time Popularities," *IEEE J. Select. Areas Commun.*, vol. 12, no. 1, pp. 180–190, 2018.

[36] S. O. Somuyiwa, A. György, and D. Gündüz, "A Reinforcement-Learning Approach to Proactive Caching in Wireless Networks," *IEEE J. Select. Areas Commun.*, vol. 36, no. 6, pp. 1331–1344, 2018.

[37] A. Sadeghi, F. Sheikholeslami, A. G. Marques, and G. B. Giannakis, "Reinforcement Learning for Adaptive Caching With Dynamic Storage Pricing," *IEEE J. Select. Areas Commun*, vol. 37, no. 10, pp. 2267–2281, 2019.

[38] L. E. Chatzieleftheriou, M. Karaliopoulos, and I. Koutsopoulos, "Jointly Optimizing Content Caching and Recommendations in Small Cell Networks," *IEEE Trans. Mobile Comput.*, vol. 18, no. 1, pp. 125–138, 2019.

[39] T. Giannakas, P. Sermpezis, and T. Spyropoulos, "Network friendly recommendations: Optimizing for long viewing sessions," *IEEE Trans. on Mobile Comput.*, 2021.

[40] Y. Fu, Y. Zhang, A. Wong, and T. Q. Quek, "Revenue maximization: The interplay between personalized bundle recommendation and wireless content caching," *IEEE Trans. on Mobile Comput.*, 2022.

[41] A. Antoniadis, C. Coester, M. Elias, A. Polak, and B. Simon, "Online Metric Algorithms with Untrusted Predictions," in *Proc. of ICML*, 2020.

[42] D. Rutten, N. Christianson, D. Mukherjee, and A. Wierman, "Online Optimization with Untrusted Predictions," 2022. [Online]. Available: https://arxiv.org/abs/2202.03519

[43] L. Andrew, S. Barman, K. Ligett, M. Lin, A. Meyerson, A. Roytman, and A. Wierman, "A tale of two metrics: Simultaneous bounds on competitiveness and regret," in *Proc. of COLT*, 2013.

[44] K. Chen, and L. Huang, "Timely-Throughput Optimal Scheduling With Prediction," *IEEE/ACM Tran. on Networking*, vol. 26, no. 6, 2018.

[45] X. Huang, S. Bian, X. Gao, W. Wu, Z. Shao, Y. Yang, J. C.S. Lui, "Online VNF Chaining and Predictive Scheduling: Optimality and Trade-Offs," *IEEE/ACM Tran. on Networking*, Early Access, 2021.

[46] Z. Zhou, X. Chen, W. Wu, D. Wu, and J. Zhang, "Predictive Online Server Provisioning for Cost-Efficient IoT Data Streaming Across Collaborative Edges," in *Proc. of ACM Mobihoc*, 2019.

[47] J. Comden, S. Yao, N. Chen, H. Xing, and Z. Liu, "Online Optimization in Cloud Resource Provisioning: Predictions, Regrets and Algorithms," in *Proc. of ACM Sigmetrics*, 2019.

[48] J. Abernethy, C. Lee, A. Sinha, and A. Tewari, "Online linear optimization via smoothing," in *Proc. of COLT*, 2014.

[49] S. Sachs, H. Hadiji, T. van Erven, and C. Guzmán, "Between stochastic and adversarial online convex optimization: Improved regret bounds via smoothness," 2022. [Online]. Available: https://arxiv.org/abs/2202.07554

[50] S. De Rooij, T. Van Erven, P. D. Grünwald, and W. M. Koolen, "Follow the leader if you can, hedge if you must," *J. Mach. Learn. Res.*, vol. 15, no. 1, p. 1281–1316, 2014.

[51] H. B. McMahan, "A Survey of Algorithms and Analysis for Adaptive Online Learning," *J. Mach. Learn. Res.*, vol. 18, no. 1, pp. 3117–3166, 2017.

[52] e. a. O. Dekel, "Online Learning with a Hint," in *Proc. of NeurIPS*, 2017.

[53] A. Bhaskara, A. Cutkosky, R. Kumar, and M. Purohit, "Online Learning with Imperfect Hints," in *Proc. of ICML*, 2020.

[54] ——, "Online learning with many hints," in *Proc. of NeurIPS*, 2020.

[55] D. Anderson, G. Iosifidis, and D. Leith, "Lazy Lagrangians with Predictions for Online Learning," *arXiv preprint arXiv:2201.02890*, 2022.

[56] A. S. Suggala and P. Netrapalli, "Online non-convex learning: Following the perturbed leader is optimal," in *Proc. of ALT*, 2020.

[57] A. Suggala and P. Netrapalli, "Follow the perturbed leader: Optimism and fast parallel algorithms for smooth minimax games," in *Proc. of NeurIPS*, 2020.

[58] N. Alon and J. H. Spencer, *The probabilistic method*. John Wiley & Sons, 2016.

[59] W. G. Madow, "On the Theory of Systematic Sampling," *The Annals of Mathematical Statistics*, vol. 20, no. 3, pp. 333–354, 1949.

[60] A. Cohen and T. Hazan, "Following the Perturbed Leader for Online Structured Learning," in *Proc. of the ICML*, 2015.

[61] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms*. MIT press, 2022.

[62] D. P. Bertsekas, "Stochastic optimization problems with nondifferentiable cost functionals," *Journal of Optimization Theory and Applications*, vol. 12, no. 2, pp. 218–231, 1973.

[63] S. Martello and P. Toth, *Knapsack Problems: Algorithms and Computer Implementations*. J. Willey & Sons, 1990.

[64] A. Kalai and S. Vempala, "Efficient algorithms for online decision problems," *Journal of Computer and System Sciences*, vol. 71, no. 3, pp. 291–307, 2005.

[65] V. V. Vazirani, *Approximation algorithms*. Springer, 2001, vol. 1.

[66] R. Gramacy, M. Warmuth, S. Brandt, and I. Ari, "Adaptive Caching by Refetching," in *Proc. of NIPS*, 2002.

[67] L. Rodriguez, F. Yusuf, S. Lyons, E. Paz, and R. Rangaswami, "Learning Cache Replacement with Cacheus," in *Proc. of USENIX Conferecne on File and Storage Technologies*, 2021.

[68] F. M. Harper and J. A. Konstan, "The MovieLens Datasets: History and Context," *ACM Trans. Interact. Intell. Syst.*, vol. 5, no. 4, 2015.

[69] G. J. Gordon, A. Greenwald, and C. Marks, "No-regret learning in convex games," in *Proceedings of the 25th international conference on Machine learning*, 2008, pp. 360–367.

[70] A. Joshi and A. Sinha, "Universal caching," *arXiv preprint arXiv:2205.04860*, 2022.

[71] K. Poularakis, G. Iosifidis, and L. Tassiulas, "Approximation Algorithms for Mobile Data Caching in Small Cell Networks," *IEEE Trans. Commun.*, vol. 62, no. 10, pp. 3665–3677, 2014.

[72] S. Mukhopadhyay, S. Sahoo, and A. Sinha, "k-experts-online policies and fundamental limits," in *International Conference on Artificial Intelligence and Statistics*.   PMLR, 2022, pp. 342–365.

[73] M. Zink, K. Suh, Y. Gu, and J. Kurose, "Characteristics of YouTube Network Traffic at a Campus Network - Measurements, Models, and Implications," *Comput. Netw.*, vol. 53, no. 4, pp. 501–514, 2009.

A APPENDIX

A.1 Comparative Summary with Related Work

Table 1 shows the performance and complexity trade-offs for the presented algorithms, and compares them to recent studies of discrete no-regret caching in the literature. The best case refers to the situation where the request predictions are perfect $\tilde{\theta}_t = \theta_t, \forall t$. The worst case refers to the situation where predictions are furthest from the truth $\tilde{\theta}_t = \arg\max_\theta \|\theta - \theta_t\|, \forall t$. The previous studies have the best and worst case columns merged as they do not utilize predictions. Furthermore, the works of [11] and [12] assume and utilize knowledge of the time horizon T ([22] uses the standard doubling trick) and use the Lipschitz constant for the gradient (i.e., request) vector. Thus, they are not classified as performing Adaptive Learning (**Adap. Learn.**) as defined by [51], which argues about the advantages of adaptive algorithms of the sort presented here. While the authors in [11] discuss the bipartite model, their simpler linear *elastic* model of utility is different than the one considered here (see [11, Sec. 3.2]). Hence, we compare to their single cache result. Finally, for algorithm 5, we make explicit the dependence on *weights* regret $R_T^{(w)}$, although it is still $R_T^{(w)} \leq O(\sqrt{T})$ to clarify the cause of inferior performance of the experts-based optimism in the worst case compared to adaptive smoothing, which even appears in the simulations.

A.2 Madow's Sampling Algorithm

Algorithm 7 describes how we obtain an integral caching vector from the continuous one. We start by sampling a uniform scalar and then loop for C iterations, including in our gradually-built set exactly one item per iteration. Hence we ensure the resulting set satisfies the capacity constraint. During an iteration, we include an item if its probability (continuous variable) falls in a carefully designed range: $[\pi_{j-1}, \pi_j]$. Hence, each item is included with probability $\pi_{j-1} - \pi_j = \hat{x}_i$. We refer the reader to [59] for futher details.

Algorithm 7: Madow's Sampling (`MadowSample`)

---

1 **Input**: $\hat{x} \in [0,1]^N, \sum x_{i \in \mathcal{N}} \leq C$.

2 **Output**: Random set S,   s.t $|S| = C$ and $\Pr(i \in S) = x_i$

3 Sample a uniformly random scalar $U \in [0,1]$

4 Define the cumulative probabilities $\pi_0 = 0$, $\pi_i = \pi_{i-1} + \hat{x}_i$, $\forall\, 1 \leq i \leq N$

5 **for** $i = 0, 1, \dots, C$ **do**

6 $\quad$ $S \leftarrow S \cup \{j : \pi_{j-1} \leq U + i < \pi_j\}$

$\quad$ **end**

7 **return** S

Table 1. Online discrete caching policies with *adversarial no regret* guarantees: a summary of the contributions and comparison with literature. For the constant α, recall that $\alpha = 1$ indicates the regular regret. Otherwise, we have α-approximate regret (see equation (16)).

Alg.	Model and Conditions	Guarantees ($R_T^{(\alpha)} \leq$)		Comput. Complex.	Approx. Const. α	Adap. Learn.
		Best case	Worst case			
1	• Single cache • Predictions	0	$O(\sqrt{T})$	$O(N^2)$	1	✓
2	• Single cache • Predictions	0	$O(poly\text{-}log(N)\sqrt{T})$	$O(N)$	1	✓
3	• Single cache • Predictions • Unequal sizes	0	$O(\sqrt{T})$	$O(N^2)$	1/2	✓
4	• Single cache • Predictions • Unequal sizes	0	$O(poly\text{-}log(N)\sqrt{T})$	$O(N)$	1/2	✓
5	• Bipartite Network • Predictions	0	$O(\sqrt{T})$	$O(N^2)$	$1 - 1/e$	✓
6	• Single Cache • Predictions	$b < 0$	$R_T^{(w)} + O(\sqrt{T})$	$O(N)$	1	✓
[11]	• Single cache	$O(poly\text{-}log(N)\sqrt{T})$		$O(N)$	1	–
[12]	• Single cache	$O\left(\sqrt{T}\right)$		$O(N)$	1	–
[16]	• Bipartite network	$O(poly\text{-}log(N)\sqrt{T})$		$O(N)$	$1 - 1/e$	✓
[22]	• General network	$O(poly\text{-}log(N)\sqrt{T})$		$O(N)$	$1 - 1/e$	–

A.3   Dependent Rounding Algorithm (*DepRound*)

The dependent rounding algorithm operates sequentially. At each iteration, it picks two continuous variables and transfers at least one of them into an integer (through the *if* statements in lines 4 to 8), while adjusting the other one (lines 9 to 12). Hence, we ensure that when the algorithm terminates, only one item is still fractional. The properties of the resulting vector listed in Lemma 2 are proved in [28, Lem. 2.1].

A.4   Proof of Theorem 4

PROOF. Since $y_{ti} \in \{0, 1\}$ $\forall t, i$, and the sampling in line 8 is uniform, we get $\mathbb{E}[y_{ti}] = \frac{1}{2}$. Hence

$$\mathbb{E}[y_{ti}] \geq \frac{1}{2}\hat{y}_{ti}. \tag{21}$$

where we have used that $\hat{y}_t \in [0, 1]^N$. Now, from the definition of 1/2-Regret we have:

$$R_T^{(1/2)} = \frac{1}{2}\sum_{t=1}^{T}\langle\theta_t, y^*\rangle - \mathbb{E}\left[\sum_{t=1}^{T}\langle\theta_t, y_t\rangle\right] \overset{(a)}{\leq} \frac{1}{2}\left(\sum_{t=1}^{T}\langle\theta_t, y^*\rangle - \sum_{t=1}^{T}\langle\theta_t, \hat{y}_t\rangle\right)$$

$$\overset{(b)}{=} 1.84\sqrt{C}\left(\ln\frac{Ne}{C}\right)^{1/4}\sqrt{\sum_{t=1}^{T}\|\theta_t - \tilde{\theta}_t\|_1^2}.$$

where inequality (*a*) follows from the 1/2-approximation property of the randomized rounding algorithm (21); and (*b*) follows from the result of Theorem[16] 3.

□

---

[16]Theorem 3 operates on integral decisions y_t. Nonetheless, even if we allow $y^\star, y_t \in \text{conv}(\mathcal{X})$, they are still integral due to the linear program in line 6 of Algorithm 2 ($\{0, 1\}$ decision variables with non-negative coefficients).

Algorithm 8: Dependent Rounding (DepRound)

---

1 **Input**: $a \in [0,1]^N, s \in R_+^N$.

2 **Output**: b satisfying points in lemma-2.

3 **while** *a contains two or more fractional elements* **do**

4   Denote the two left most fracitonal elements a_i and a_j.

5   **if** $0 \le s_i a_i + s_j a_j \le \min\{a_i, a_j\}$ **then**

         | Set $b_i = 0$ with probability $s_j a_j / s_i a_i + s_j a_j$. With the remaining probability set $b_j = 0$

     end

6   **if** $a_i \le s_i a_i + s_j a_j \le a_j$ **then**

         | Set $a_i = 1$ with probability a_i. With the remaining probability set $a_i = 0$

     end

7   **if** $a_j \le s_i a_i + s_j a_j \le a_i$ **then**

         | Set $a_j = 1$ with probability a_j. With the remaining probability set $a_j = 0$

     end

8   **if** $\max\{a_i, a_j\} \le s_i a_i + s_j a_j \le a_i + a_j$ **then**

         | Set $b_i = 1$ with probability $s_j(1-a_j) / (s_i(1-a_i)) + (s_j(1-a_j))$. With the remaining set $b_j = 1$

     end

9   if $b_i = 0$ set $b_j = s_i/s_j\, a_i + a_j$

10   if $b_i = 1$ set $b_j = a_j - s_i/s_j\, (1 - a_i)$

11   if $b_j = 0$ set $b_i = a_i + s_j/s_i\, a_j$

12   if $b_j = 1$ set $b_i = a_i - s_j/s_i\, (1 - a_j)$

     end

     return b

A.5 Proof of Theorem 5

First, we show that the $1/2$-point-wise approximation holds for $\{x_t\}_t$. Then, we re-use the result of Theorem 2. In detail, by Lemma 2, the DepRoundsubroutine returns $\bar{x}_t$ such that $\mathbb{E}[\bar{x}_t] = \hat{x}_t$. Then, by the same argument about uniform sampling in the proof of Theorem 4, we have that $\mathbb{E}[x_t] \ge \frac{1}{2}\bar{x}_t$, where $x_t \in \mathcal{X}_s$. We recover our $1/2$-approximation guarantee for OFTRL iterates $\{x_t\}_t$:

$$\mathbb{E}[x_t] \ge \frac{1}{2}\hat{x}_t. \tag{22}$$

By the definition of $1/2$-regret guarantee, we have

$$R_T^{(1/2)} = \frac{1}{2}\sum_{t=1}^{T}\langle \theta_t, \hat{x}^* \rangle - \mathbb{E}\left[\sum_{t=1}^{T}\langle \theta_t, x_t \rangle\right] \overset{(a)}{\le} \frac{1}{2}\Big(\sum_{t=1}^{T}\langle \theta_t, \hat{x}^* \rangle - \sum_{t=1}^{T}\langle \theta_t, \hat{x}_t \rangle\Big) \overset{(b)}{=} \sqrt{C}\sqrt{\sum_{t=1}^{T}\|\theta_t - \tilde{\theta}_t\|_2^2}.$$

where inequality (a) follows from the $1/2$-approximation property of the randomized rounding algorithm (22); and (b) follows from the result of Theorem 2. In fact, that theorem operates on integral decisions x_t, yet we can recall that $x^\star$ is integral and that we showed $\mathbb{E}[x_t] = \hat{x}_t$; thus, the same guarantee can be used here.

A.6 Proof of Theorem 6

We start from the result of [15, Thm. 1] (or its earlier version from [17]), which provides guarantees for the regret of the continuous variables $\hat{x}$. For the moment, assume that we have the following point-wise approximation for the decision variables $\mathbb{E}[\hat{k}] = k$, $\mathbb{E}[\hat{u}] \ge (1 - 1/e)u$, and that the capacity constraints are respected. Then, due to the linearity of the objective function $f_t(\cdot)$, we get

that our α-regret (with $\alpha = 1 - 1/e$) is:

$$R_T^{(1-1/e)} = 1 - \frac{1}{e} \sum_{t=1}^{T} \langle \theta_t, \hat{x}^* \rangle - \mathbb{E} \sum_{t=1}^{T} \langle \theta_t, x_t \rangle \overset{(a)}{\leq} 1 - \frac{1}{e} \Big(\sum_{t=1}^{T} \langle \theta_t, \hat{x}^* \rangle - \sum_{t=1}^{T} \langle \theta_t, \hat{x}_t \rangle \Big)$$

$$\overset{(b)}{=} (2 - \frac{2}{e}) \sqrt{1 + JC} \sqrt{\sum_{t=1}^{T} \|\theta_t - \tilde{\theta}_t\|_2^2}.$$

where inequality (a) follows from the $(1 - 1/e)$-approximation property of the randomized rounding algorithm; and (b) follows from the result of Theorem [15, Thm. 1].

Now, to show that $\mathbb{E}[\hat{k}] = k$, we follow the same argument in the proof of Theorem 2. Namely, due to Madow's sampling, each file is included with probability $\hat{k}$, and at most C_j files are included at each cache. Hence, $\mathbb{E}\left[k_{nj}\right] = \hat{k}_{nj}, \forall n, j$. Regarding the point-wise approximation for u, we define the set $\mathcal{J}^i$ of caches connected to user i:

$$\mathcal{J}^i = \left\{ j \in \mathcal{J} \,\middle|\, d_{ij} = 1 \right\}.$$

Then, we have

$$\mathbb{E}\left[u_{nij}\right] \overset{(a)}{=} \Pr[u_{nij} = 1] \overset{(b)}{=} \Pr[\vee_{j \in \mathcal{J}^i} k_{nj} = 1] \overset{(c)}{=} 1 - \prod_{j \in \mathcal{J}^i} \left(1 - \hat{k}_{nj}\right)$$

$$\overset{(d)}{\geq} 1 - e^{-\sum_{j \in \mathcal{J}^i} \hat{k}_{nj}} \overset{(e)}{\geq} 1 - e^{-\hat{u}_{nij}} \overset{(f)}{\geq} \left(1 - \frac{1}{e}\right) \hat{u}_{nij}.$$

Where in the above chain of inequalities, (a) follows from u_{nij} being are binary variable; (b) by the construction of the algorithm (step 8); and (c) from the independent rounding for each cache. Also, inequality (d) follows from from $e^x \geq 1 + x, \forall x \in \mathbb{R}$; (e) from the relaxed version of the caching/routing constraint of $\mathbb{P}_2$, and finally (f) from the concavity of $1 - e^{-x}$ and the domain of $\hat{u}_{nij}$ being restricted to $[0, 1]$.

A.7   Additional Simulations

A.7.1   Probabilistic Predictions. Let us first demonstrate, with a simple example, that using probabilistic predictions is beneficial for the performance of optimistic algorithms. The regret bound of the proposed algorithms depend on the terms $\|\theta_t - \tilde{\theta}_t\|, \forall t$. Now, consider a prediction $\tilde{\theta}_t$ that places ϵ probability mass on the correct file, and the remaining uniformly over the rest of the files in the library. Then, we get:

$$\|\theta_t - \tilde{\theta}_t\|_2 \approx 1 - \epsilon, \quad \text{since} \quad \frac{(1 - \epsilon)^2}{(N - 1)} \approx 0,$$

compared to a mis-prediction (or, mistaken) one-hot $\tilde{\theta}_t$ which will have $\|\theta_t - \tilde{\theta}_t\|_2 = \sqrt{2}$. Using the ℓ_1 norm, a one-hot mistake costs 2 compared to $2 - 2\epsilon$ for the probabilistic one. We stress again that all the results presented in this work hold both for probabilistic and for deterministic predictions. The former can be taken directly from the output of a forecasting model, while one can create the latter by simply using the highest-probability request.

We continue by presenting experimental results for a probabilistic prediction model with varying accuracy. In detail, in Fig. 4 we measure the regret of the proposed OFTRL and OFTPL policies after $5k$ time steps (i.e., file request) using the well-known YouTube request trace [73] with $N = 10^4$ and $C = 150$. R_{5k} is measured using prediction vectors with varying density that is placed on the file to
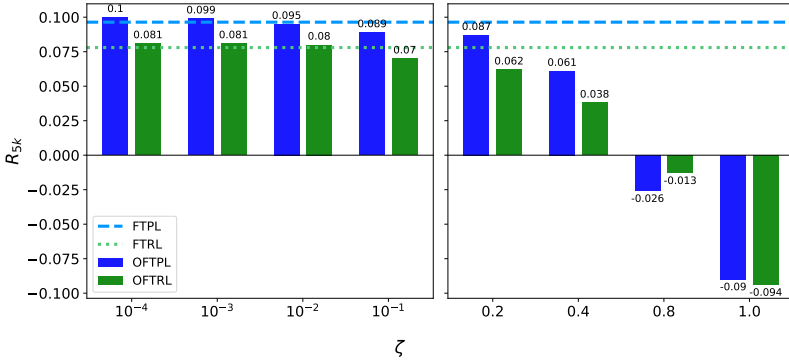
Fig. 4. Regret with varying probability mass placed on the correct file in the prediction vector.

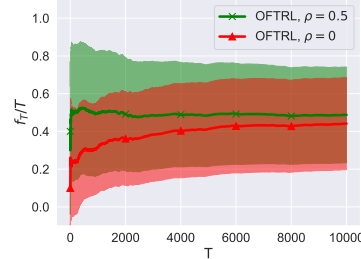Fig. 5. Average regret of experts-based optimism.

Fig. 6. Average OFTRL hit-rate for cache network.

be requested. Namely, if at step t, the requested file is n, we feed the optimistic algorithms with a prediction vector:

$$\tilde{\theta} \quad \text{with:} \quad \tilde{\theta}_n^t = \zeta \quad \text{and} \quad \tilde{\theta}_{n'}^t = (1 - \zeta)/(N - 1), \ \forall n' \neq n.$$

That is, the prediction vector has ζ probability placed on the file to be requested, and the remaining $(1 - \zeta)$ uniformly distributed across the remaining files. We note that when the prediction vector is almost uniform (i.e., $\tilde{\theta}$ contains no useful information), the optimistic versions nearly match the non optimistic ones.

We can see that at $\zeta = 0.1$, both OFTRL and OFTPL already start outperforming their non-optimistic counterparts, by 9.6% and 7.8%, respectively. Also, they outperform the best-in-hindsight benchmark $x^\star$ when the accuracy becomes reasonably high ($\zeta \geq 0.8$). Lastly, we see that OFTRL has a performance advantage of up to 28.7% compared to OFTPL, when fed the same predictions, at the expense of its additional computation complexity (see next section).

A.7.2 Algorithms 5 and 6. Fig. 5 plots the regret for the expert-based Algorithm 6. Note that the R_T can reach negative values, i.e., outperform better the benchmark, when $\rho = 0.5$. This is aligned with the bound in (20) and hints to the fact that stronger benchmarks can be used for this algorithm. However, it performs worse than the regularization-based optimism in the case where $\rho = 0$, achieving regret $R_T = 0.143$ at time $T = 4k$ compared to $R_T = 0.075$ (OFTRL) and $R_T = 0.107$ (OFTPL). Lastly, the bipartite *utility* is shown in Fig. 6, the *hit-ratio* of OFTRL is approximately 0.49 when $\rho = 0.5$. Expectantly, the performance drops when $\rho = 0$, but steadily increases from 0.30 at $T = 500$, to 0.44 at $T = 10k$.