Optimistic Learning with Applications to Caching Networks

Mhaisen, N.

DOI
10.4233/uuid:68fd39be-e3d5-4b3a-a3e5-76c80b3121f0

Publication date
2025

Document Version
Final published version

Citation (APA)
Mhaisen, N. (2025). *Optimistic Learning with Applications to Caching Networks*. [Dissertation (TU Delft), Delft University of Technology]. https://doi.org/10.4233/uuid:68fd39be-e3d5-4b3a-a3e5-76c80b3121f0

Optimistic Learning

with Applications to Caching Networks

Optimistic Learning

with Applications to Caching Networks

Dissertation

for the purpose of obtaining the degree of doctor
at Delft University of Technology,
by the authority of the Rector Magnificus, prof.dr.ir. T.H.J.J. van der Hagen;
Chair of the Board for Doctorates
to be defended publicly on Monday 24 November 2025 at 17:30 o'clock.

by

Naram MHAISEN

Master of Science in Computing,
Qatar University, Qatar.
Born in Damascus, Syria.

This dissertation has been approved by the promotors
Prof. dr. K.G. Langendoen
Dr. G. Iosifidis

Composition of the doctoral committee:

Rector Magnificus,	Chairman
Prof. dr. K.G. Langendoen,	Delft University of Technology, promoter.
Dr. G. Iosifidis,	Delft University of Technology, copromoter.
Independent members:	
Prof. dr. G. Dán	KTH Royal Institute of Technology, Sweden.
Prof. dr. P. Grosso	University of Amsterdam.
Prof. dr. M.T.J Spaan	Delft University of Technology.
Prof. dr. T. Spyropoulos	Technical University of Crete, Greece.
Dr. T.A.L. van Erven	University of Amsterdam.

Keywords: online learning, network optimization.

ISBN 000-00-0000-000-0

An electronic version of this dissertation is available at
`http://repository.tudelft.nl/`.

Contents

Summary

AI/ML-based approaches are at the forefront of resource management in modern communication networks. Deep learning, in particular, enables fast and high-performing decision-making when sufficient representative training data is available to build accurate offline models. Conversely, online learning solutions operate without prior training and make decisions based on real-time observations; however, they tend to be overly conservative to ensure robustness (i.e., worst-case guarantees).

This thesis advocates *optimistic learning* as a decision-making framework for resource management in networked systems. An optimistic learning algorithm integrates untrusted predictions and assesses their accuracy at runtime. When predictions are accurate, these algorithms achieve performance levels comparable to offline-trained models. Crucially, they maintain the robustness of regular online learning, ensuring reliability even when predictions are inaccurate.

We focus on caching networks and propose new optimistic learning algorithms for *coded caching*, and *whole-file* caching. These algorithms provably converge to the best fixed caching allocation at an order-optimal rate, independent of prediction accuracy. However, when predictions are accurate, convergence is highly accelerated, achieving the "optimistic" premise.

We then extend our focus to scenarios where the optimization target itself *changes* over time. In caching, this translates to competing against dynamic caching configurations rather than a single best fixed allocation. We demonstrate that optimism is even more valuable in this setting; accurate predictions help the learner efficiently track moving targets, adapting in real-time without excessive conservatism. Furthermore, we explore the role of predictions in stateful systems, where past decisions influence future costs. In such environments, optimistic learning benefits from *horizon-based* predictions, leveraging forecasts over extended time windows rather than immediate next-cost predictions.

All proposed algorithms are rigorously analyzed and come with provable performance guarantees under carefully designed and explicitly stated metrics. By integrating optimistic learning into network optimization, this thesis explores the spectrum between prediction-driven and robust approaches, offering a principled framework for leveraging untrusted ML predictions in network resource allocation.

Samenvatting

AI/ML-gebaseerde benaderingen staan aan de voorhoede van resource-management in moderne communicatienetwerken. Deep learning, in het bijzonder, maakt snelle en zeer performante besluitvorming mogelijk wanneer er voldoende representatieve trainingsdata beschikbaar is om nauwkeurige offline-modellen te bouwen. Omgekeerd werken online-leeralgoritmen zonder voorafgaande training en nemen zij beslissingen op basis van realtime-observaties; zij zijn echter vaak te conservatief om robuustheid te garanderen (d.w.z. garanties in het slechtste geval).

Dit proefschrift bepleit *optimistisch leren* als besluitvormingskader voor resource-management in netwerksystemen. Een optimistisch leeralgoritme integreert onbetrouwbare voorspellingen en evalueert hun nauwkeurigheid tijdens de uitvoering. Wanneer de voorspellingen accuraat zijn, behalen deze algoritmen prestatieniveaus die vergelijkbaar zijn met offline getrainde modellen. Cruciaal is dat zij de robuustheid van regulier online leren behouden, zodat de betrouwbaarheid ook bij onnauwkeurige voorspellingen gewaarborgd blijft.

Wij richten ons op caching-netwerken en introduceren nieuwe optimistische leeralgoritmen voor *coded caching* en *whole-file caching*. Deze algoritmen convergeren aantoonbaar met een orde-optimale snelheid naar de beste vaste cache-allocatie, onafhankelijk van de voorspellingsnauwkeurigheid. Wanneer de voorspellingen wel accuraat zijn, verloopt de convergentie echter aanzienlijk sneller en wordt zo de öptimistische"belofte gerealiseerd.

Daarna verbreden wij onze focus naar scenario's waarin het optimalisatiedoel zelf in de tijd *verandert*. In caching komt dit neer op het concurreren met dynamische cache-configuraties in plaats van één beste vaste allocatie. Wij tonen aan dat optimisme in deze context nog waardevoller is: nauwkeurige voorspellingen stellen de learner in staat om bewegende doelwitten efficiënt te volgen en zich in realtime aan te passen zonder buitensporige conservativiteit. Verder onderzoeken wij de rol van voorspellingen in stateful systemen, waar beslissingen uit het verleden de toekomstige kosten beïnvloeden. In dergelijke omgevingen profiteert optimistisch leren van *horizon-gebaseerde* voorspellingen, waarbij gebruik wordt gemaakt van prognoses over langere tijdvensters in plaats van louter de eerstvolgende kost.

Alle voorgestelde algoritmen zijn grondig geanalyseerd en voorzien van formele prestatiegaranties onder zorgvuldig ontworpen en expliciet beschreven maatstaven. Door optimistisch leren in netwerkoptimalisatie te integreren, verkent dit proefschrift het spanningsveld tussen voorspellingsgestuurde en robuuste benaderingen en biedt het een principieel kader om onbetrouwbare ML-voorspellingen te benutten bij het alloceren van netwerkbronnen.

1

Introduction

This dissertation focuses on the development and application of principled optimization and machine learning techniques to resource allocation problems in communication networks. Specifically, we adopt the framework of *online learning* and apply it to the fundamental problem of *caching*. We explore different variations of the online learning paradigm suited for different versions of the caching problem.

At a high level, online learning provides a formal approach to sequential decision-making and offers algorithms with performance guarantees on those decisions. In this setting, a learner (i.e., an algorithm) makes decisions over a sequence of time slots, which could range from milliseconds to hours or even days, depending on the system specifications. At each decision slot, the learner selects an action, which, in our case, corresponds to a resource allocation decision. Only after committing to this action does the learner get to see how good (or bad) it has been. While the learner can use historical performance data about *past* decisions, it must make the *current* decision without knowing how well it will perform in *this* time slot. This paradigm naturally captures many problems in networking. Consider the following problem of *caching*: at each time slot, the goal is to decide which files to store close to users so that they get delivered faster when they are requested. The challenge is that future user requests are unknown. While past access patterns provide useful signals, the decision must ultimately be made before the next set of requests is known (i.e., performed). This classical problem was indeed studied under different assumptions on the requests, which are being challenged in today's networks. Equipped with the online learning machinery, we aim to design caching algorithms that achieve optimal "hit" ratios with *no assumptions* on the request pattern.

Optimism. Now, consider a subtle yet powerful twist: what if we have *predictions* of future requests? At first glance, incorporating predictions might seem to simplify the problem by reducing uncertainty. This would indeed be the case if the predictions were reasonably accurate. However, real-world predictions are often imperfect, and relying on inaccurate forecasts can lead to even more suboptimal decisions, effectively increasing uncertainty rather than reducing it. A natural re-

sponse might be to only incorporate the predictions up to a controllable degree, ensuring decisions remain primarily based on past observation. While this cautious approach mitigates potential risks, it also restricts potential gains when predictions are, in fact, reliable. This trade-off naturally gives rise to a fundamental dilemma:

- Robustness: When the predictions are *inaccurate*, how can one ensure that the performance remains close to what would have been achieved *without them*?
- Prediction-adaptivity: When the predictions are *accurate*, how can one ensure that the performance remains close to what would have been achieved *by relying on them*?

Note that the most robust algorithm would be one that entirely discards predictions, preventing inaccurate forecasts from degrading performance. Conversely, the most prediction-adaptive algorithm would simply follow the predictions, fully capitalizing on their accuracy. However, either extreme fails when its underlying assumption is violated; ignoring predictions sacrifices potential benefits, while "blindly" following them risks significant losses. Our objective is to design best-of-both-worlds algorithms that, in hindsight, perform competitively with either extreme depending on the *actual* quality of predictions.

Balancing the trade-off in the above two bullet points is at the core of this thesis, and we refer to algorithms that (optimally) balance this trade-off as **optimistic learning** algorithms, or optimistic algorithms for short. The term "optimistic" stems from the hope that if predictions are accurate, one can achieve significant gains (prediction-adaptivity), while if they are inaccurate, the losses can remain minimal (robustness).

The next section traces the evolution of resource allocation in networked systems, from classical optimization formulations to modern learning-based approaches. We then transition from the offline flavor of this learning approach to the online one, which naturally leads us to aim for a best-of-both-worlds approach captured by the optimistic learning framework. Following this, we formally introduce optimistic online learning, outline its origins, and set the stage for the specific problems addressed in this thesis.

1.1. Facets of Network Control: from Optimization to Learning

The framework of Network Utility Maximization (NUM) was the outcome of a systematic effort to create a general toolbox for optimizing communication systems [1, 2]. In NUM, the network controller (NC) has prior access to user demands and system parameters (e.g., link delays), and formulates an optimization problem that defines the desired system operation, including the optimization criteria (e.g., throughput) and its operational and resource constraints. The problem is solved *offline* and the system is then operated based on the obtained solution. NUM has been developed using convex optimization models and algorithms [3], and its optimality guarantees (due to convexity) and decomposability [4] have rendered it a

powerful tool for designing system architectures and protocols, including a cross-layer approach [5].

The *stochastic* NUM (SNUM) framework[1] [7] extended this methodology to dynamic systems where the user demands and system parameters vary with time, based on some stationary random process. In this case, instead of solving an offline problem and applying its solution one-off, the NC makes decisions $\boldsymbol{x}_t$ in a time-slotted fashion after observing the system state at the start of each slot t. The seminal Max-weight and Back-pressure policies [6], and their Drift-plus-Penalty extensions [8], guarantee optimal performance and stability of the involved queues (e.g., backlog of requests), while being oblivious to the statistics of the perturbations. In effect, these policies ensure performance commensurate with that of an ideal oracle policy $\boldsymbol{x}^\star$ with access to all future system and user parameters. SNUM has been instrumental in optimizing wireless networks [9] and various other systems (e.g., smart grid), that are modeled as networks of queues [10].

Despite its success, there is growing consensus that (S)NUM cannot serve as the primary optimization toolbox for future communication networks. These networks will be significantly larger and more complex than those for which (S)NUM was originally aimed. In particular, the common assumption that user and system parameters are either known a priori or evolve according to a stationary process is increasingly impractical. For example, in small cell wireless networks, user churn is non-stationary and often unpredictable [11]; and virtualized base stations in mobile networks exhibit platform and data dependent throughput and energy consumption, which cannot be modeled accurately [12]. More broadly, future networks must support a significantly larger and more diverse user base, encompassing cyber-physical systems (e.g., robots and autonomous vehicles), IoT nodes, and embedded devices, each with distinct service requirements and resource constraints [13]. The limitations of (S)NUM become even more pronounced with the increasing softwarization of networks, where critical functions are deployed on virtualized computing platforms with inherently *volatile* and hard-to-predict resource-sharing dynamics [14].

In more technical terms, there is a lack of information on the values of the various user and system parameters, making the actual utility functions connecting these parameters with the performance metrics of interest unknown to the NC. From a network management perspective, this change has immense implications. Essentially, it turns the various network optimization problems that the NC needs to tackle, into learning problems where it needs to devise decisions under information asymmetry, namely to decide $\boldsymbol{x}_t$ without knowing the cost function $f_t(\boldsymbol{x})$, while the user dynamics and other system perturbations are highly volatile. This calls for the development of novel learning-based NUM tools.

1.2. Offline, Online & Optimistic Learning

Recently, Machine Learning (ML), and in particular *Deep Learning* (DL), has emerged as a promising approach for network control and resource management [15]. Lever-

[1] Some queuing control policies for networks, e.g., [6], have in fact preceded the development of NUM tools.

aging the abundance of raw measurement data in these systems, DL can automate the prediction of future parameter values (e.g., channel gains) [16] and enable the network controller to recover the objective functions $f_t(\boldsymbol{x})$ that need to be optimized in each time slot t. Additionally, DL can directly generate control decisions $\boldsymbol{x}_t$ by solving large-scale problems in near real-time [17]. Indeed, DL and ML in general, have been proposed as replacements for traditional optimization techniques in network management [18] and for addressing specific problems such as traffic engineering [19], the design of intelligent services [20], and the optimization of PHY-layer communications [21], among others. However, the effectiveness of these solutions depends on the availability of representative datasets. This is not always feasible, either because collecting such data is costly or because the problem is dynamic and non-stationary. These issues render the typical ML training cycle prohibitively slow and resource-intensive.

At the other end of the spectrum of AI-based optimization tools lies the paradigm of *online learning*, which does not require pre-processing or training. Instead, it adapts at runtime to the system and environment conditions using real-time observations. In specific, learning algorithms that rely on *Online Convex Optimization* (OCO) tools [22] are principled and provide guarantees for the performance of the online (i.e., sequential) decisions. In this case, the system operation is modeled as an online learning process over T slots, where the NC commits its decision $\boldsymbol{x}_t$ at the *start* of each slot t; observes the outcome (function $f_t(\cdot)$) at the end of the slot; and updates its strategy for coming up with $\boldsymbol{x}_{t+1}$ accordingly. OCO algorithms ensure that the performance achieved by the set of all actions $\{\boldsymbol{x}_t\}_{t=1}^{T}$ approaches gradually that of the ideal (but unknown) benchmark $\boldsymbol{x}^\star$. The benchmark can be, for example, the minimizer of the sum of the functions ($\boldsymbol{x}^\star = \arg\min_{\boldsymbol{x}} \sum_t f_t(\boldsymbol{x})$). This is formally captured using the metric of *regret* $\mathcal{R}_T$, which quantifies the gap between the cumulative cost of the learner's actions $\sum_t f_t(\boldsymbol{x}_t)$, and the cost of a benchmark $\sum_t f_t(\boldsymbol{x}^\star)$. The goal is to establish upper bounds on this gap that *diminish* as the learning horizon T increases: $\lim_{T\to\infty} \mathcal{R}_T/T = 0$, which is equivalent to saying that $\mathcal{R}_T = \mathcal{O}(T^\alpha)$ for some $\alpha < 1$. Ideally, this decay is rapid (i.e., α is as small as possible).

OCO is quite appealing from a network management perspective [23, 24] for several reasons. First, it builds on online versions of seminal algorithms like gradient descent, which have underpinned previous NUM frameworks; thereby, it inherits key properties such as optimality, decomposability, and scalability. Second, it is transparent and interpretable; namely, the regret bounds explicitly reveal how various system parameters influence $\mathcal{R}_T$. Finally, OCO ensures performance guarantees across diverse perturbation models, including adversarial settings, rendering it a versatile tool. However, this robustness comes at a cost: it relies on inherently cautious learning, treating the function landscape as entirely unknown and optimizing for worst-case scenarios. While this conservatism safeguards performance in adversarial conditions, it can lead to unnecessarily slow adaptation in more predictable scenarios. In fact, the NC often has at least short-term foresight into system and user demands, making such extreme caution unnecessarily restrictive. Put differently, while OCO's regret bounds hold universally, its learning dynamics may lag

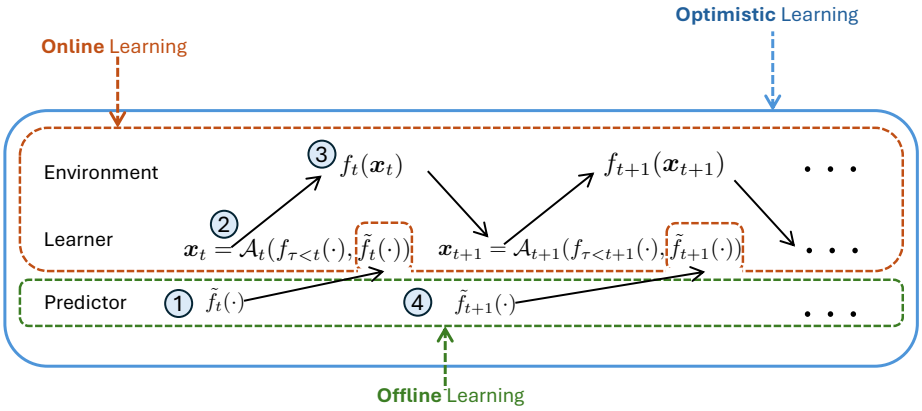


Figure 1.1: The protocol of optimistic learning: at each time slot, an untrusted prediction is first generated. Second, the learner takes an action via an algorithm $\mathcal{A}_t$, which considers past costs ($f_{\tau<t}(\cdot)$), the predicted future cost ($\tilde{f}_{t+1}(\cdot)$), and the predictions accuracy so far (implied by the subscript in $\mathcal{A}_t$). Third, the cost of the committed action is realized. A new slot then begins with another untrusted prediction. The learner's total loss is the sum of $f_t(\boldsymbol{x}_t)$. The difference between this sum and that of a *benchmark* is the regret.

behind optimal adaptation in scenarios where the problem structure allows for more aggressive, performance-efficient decisions.

Given the complementary strengths and limitations of offline and online learning, a natural question from a network management perspective is whether we can develop a framework that combines their benefits *without* inheriting their drawbacks. This thesis argues that *optimistic learning* provides a compelling answer, achieving the best of both worlds by integrating offline-trained predictors (either for functions or actions) into online learning algorithms.

A visualization of the three learning modes discussed in this chapter is presented in Fig. 1.1. Consider the offline learning setting (green outline). Here, assuming sufficient representative data has been collected, the cost functions can be predicted, and algorithm $\mathcal{A}$ selects the decisions $\boldsymbol{x}_t$ accordingly. In contrast, in the online learning setting (red outline), no predictions of future costs are used. Instead, the learner determines actions cautiously based solely on historical cost observations. For example, if past costs have exhibited high volatility, online learning algorithms typically favor the safest action, regardless of any predicted future costs. Finally, the optimistic learning paradigm (blue outline) aims to unify the benefits of both approaches. It seeks to leverage predictions while maintaining robustness. If the learner's algorithm $\mathcal{A}$ successfully balances the trade-off between prediction-adaptivity and robustness, it qualifies as an optimistic learning algorithm.

As alluded to before, the key advantage of optimistic learning is its ability to accelerate convergence to benchmark performance when predictions are accurate, while maintaining the robustness of traditional OCO methods in cases where predictions are unreliable (i.e., balance the trade-off). This adaptability is particularly

well-suited for network management, where prior training data is often available but may not always generalize perfectly to real-time conditions. Crucially, optimistic learning does not depend blindly on such prior models; instead, it evaluates their reliability using real-time observations and seamlessly transitions to pure online learning when necessary. This versatility allows NCs to deploy solutions that perform efficiently across diverse conditions, eliminating the need for rigid trade-offs between robustness and performance.

The optimistic learning framework, in its current prevalent form, originated in [25], which coined the term and introduced the early variants of optimistic algorithms. Then, [26] provided a more general treatment and further "tuning" techniques. Optimistic algorithms have since undergone multiple improvements in terms of either efficiency or performance bounds [27] and we refer the reader to [28, Sec. 7.12] for a comprehensive overview. The specific technical contributions of this dissertation, in contrast to these works and their follow-ups, will be presented in Sec. 1.5 and the introductions of the respective chapters.

1.3. Caching: a Ubiquitous Computing Problem

This thesis focuses on the 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 optimizes user experience in Content Distribution Networks (CDNs) [29], and enhances the performance of wireless networks [30]. With the perpetual growth of Internet traffic fueled by new services such as AR/VR [31], caching policies that learn fast to maximize cache hits can mitigate the increasing costs of information transportation [32], and similar benefits can be expected for embedded and other computing systems [33]. This work aspires to advance the 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*.

Beyond its evident importance in networking, the caching problem is representative of many resource allocation problems that arise across computing. Namely, caching can be framed as an instance of a broader computational problem known as the k-set selection problem (cf. [34]). Here, at each decision slot, the learner must select a set of k coordinates (objects) out of d. When $k = 1$, this reduces to the classical "expert's" problem. This formulation has applications in various domains, including online ad placement and personalized news recommendations. In the latter, for example, a website must select k news topics from a pool of d categories, such as current events, economics, and foreign affairs, to display to a user. Based on user feedback, the system must continuously adjust its topic selection to maximize engagement. Furthermore, in one form of caching (whole files with different sizes), the caching problem generalizes even further, becoming an instance of the well-known knapsack problem [35, Sec. 16.5]. This problem has extensive applications even beyond computing and networking. In other words, while caching serves as the primary application domain in this thesis, the underlying problem formulation and algorithmic solutions have far-reaching implications.

Due to its importance and ubiquity, the quest for efficient data caching policies

spans more than 50 years and remains today one of the most important research areas for communication systems [32]. Caching was first studied in computer systems where the aim was to decide which files to store in fast-accessible memory segments (*paging*) [36]. Its scope was later expanded due to the explosion of Internet [37] and the advent of CDNs [38], and was recently revisited as a technique to improve the operation of wireless networks through edge caches [30] and on-device caching [39]. A common challenge in these systems is to design an online policy that decides which files to store at a cache, without knowing the future file requests, so as to maximize the cache *hits* or other cache-related performance metric.

Classical & ML solutions for caching. There is a range of online caching policies that tackle this problem under different assumptions on the request arrivals. Policies such as LFU and LRU are widely deployed [40, 41, 42]. Under certain statistical assumptions on the request trace, such policies maintain the cache at an optimal state, see [43, Sec. 3.1-3.2]. However, with the frequent addition of new content to libraries of online services and the high volatility of file popularity [44], these policies can perform arbitrarily badly. This motivated modeling non-stationary request patterns [45, 46] and optimizing accordingly the caching decisions [47, 48]. Another line of work relies on learning techniques such as (multi-agent) reinforcement learning to estimate the request probabilities and make caching decisions accordingly [49, 50, 51]; but typically these solutions either do not offer optimality bounds, or do not scale due to having the library size in their bounds.

On the other hand, forecasting models have been used to optimize caching through, e.g., evicting the file with the furthest predicted request [52]. While these policies have shown performance gains, their hit ratio can deteriorate if the forecasted requests cease to meet reality due to a shift in the data (users requests). Follow-up works attempted to remedy this issue by designing mechanisms that identify performance deterioration and trigger re-training of the model [53]. Still, algorithms with explicit formal guarantees that are agnostic to the quality of the predictions are yet to be proposed.

Online Learning for caching. Caching was studied within the framework of online learning in [54] for a single-cache system; and in its more general form recently in [55] that proposed an online gradient descent (OGD) caching policy. Interesting follow-up works include sub-modular policies [56], online mirror-descent policies [57], and the characterization of their performance limits [58, 59]. The advantage of these online learning-based caching policies is that they are scalable, do not require training data, and their performance bounds are *robust* to any possible request pattern, even when the requests are generated by an adversary that aims to degrade the caching operation. 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 aggressively optimize the cache. For instance, requests in services like Facebook are often amenable to accurate forecasts; while in YouTube and Netflix the viewers receive recommendations that can effectively serve as predictions for their forthcoming requests [60, 61]. Unfortu-

nately, online learning-based caching policies, such as [58, 62, 63, 64, 65, 66], are *pessimistically* designed for the worst-case request sequence and cannot benefit from predictable requests. Hence, an aspect that remains hitherto unexplored is whether predictions on future requests can improve the performance of such learning-based caching policies *without sacrificing their robustness*.

Caching with (un)trusted predictions. While the interplay between predictions and caching has attracted attention from both machine learning and networking communities, direct or indirect assumptions on the quality of the predictions are often assumed [67, 68, 69]. On the other hand, [70, 71] presented a mechanism that uses untrusted predictions to achieve "competitive-ratio" guarantees. Their approach was generalized to metrical task systems by [72] and improved with nearly lower-bound matching for the competitive ratios in [73]. However, as proved in [74], algorithms that ensure constant competitive ratios do not necessarily guarantee sublinear regret, which is the performance criterion we employ here following the recent regret-based caching research [58, 62, 63, 64, 65, 66, 75]. 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 [76, 77], which assumes the requests and system perturbations are *stationary*; and in online learning [78, 79], which *does not adapt* to predictions' accuracy (considered known). In this thesis, we make *no assumptions* on the predictions' quality, which can be even adversarial.

1.4. Problem Statement

We provided a high-level overview of the online learning framework and its relevance to network resource allocation, particularly in caching. A key focus of this thesis is the challenges associated with the *optimistic* variant of this framework. This setting introduces a fundamental trade-off in algorithm design: leveraging predictions to improve performance, while ensuring robustness under all conditions. Effectively balancing this trade-off raises the following central research question:

How can sequential decision-making algorithms for (caching) networks leverage untrusted predictions while still maintaining worst-case guarantees?

While this dissertation does not fully resolve this question given the numerous variations of sequential decision-making formulations, it does address several fundamental aspects. In the following, we summarize and outline these key aspects.

1.5. Thesis Contributions and Outline

In this section, we provide a brief overview of each chapter. Each subsection outlines the problem addressed, the proposed solutions, and the key contributions.

1.5.1. Optimistic Learning for Continuous Domains: Coded Caching

We begin by demonstrating the effectiveness of optimistic learning in the context of (elastic) bipartite caching networks. In this setting, a collection of cache (edge)

servers serves a set of users or user groups. Building on the seminal "Optimistic Follow The Regularized Leader (OFTRL)" algorithm [26], we develop an optimistic online caching system that achieves state-of-the-art performance across established datasets and benchmarks. The contributions in this chapter are both technical and conceptual. On the technical front, we introduce a modified variant of OFTRL that improves the performance guarantees over the standard formulation that appears in [26]. Conceptually, we establish a formal connection between bipartite caching and online learning, framing caching as a sequential decision-making problem.

Additionally, we propose the first optimistic meta-learning framework designed to handle scenarios with *multiple* prediction sources. Rather than relying on a single predictor, our approach simultaneously learns both the most reliable predictor and the optimal caching decisions. While a similar approach was originally introduced in [80], our framework extends this idea by incorporating optimism at the meta-learning level as well. That is, attempting to predict the *accuracy* of each predictor. By dynamically adapting to the best available prediction source, our method enhances the benefits of optimistic learning beyond the single-predictor setting. As in the single-predictor case, our system consistently outperforms existing benchmarks on standardized datasets. Overall, this chapter highlights the potential of optimistic learning in a classical bipartite cache network problem, demonstrating its ability to enhance caching performance in uncertain environments.

A key technical assumption in this chapter is that files can be stored in arbitrarily small fractions. This assumption aligns with the coded-caching literature, where files are typically divided into very small chunks, enabling fractional caching. While this is neither a new nor particularly strong assumption, it facilitates mathematical tractability and algorithmic design. However, in many practical scenarios, this assumption does not hold, and caching decisions are inherently discrete—i.e., a file is either fully stored or not, and a user is either served or not. The next chapter addresses this more restrictive, but practically relevant setting, where resource allocation decisions are inherently combinatorial.

1.5.2. Optimistic Learning for Discrete Domains: Whole-file Caching

In this chapter, we shift our focus to caching networks under the whole-file constraint. This seemingly simple modification fundamentally alters the problem's structure. Specifically, we show that whole-file caching can be framed as an instance of a broader computational problem known as the general k-set selection problem (see Sec. 1.3). Furthermore, if we allow files to have different sizes, the caching problem generalizes even further, becoming an instance of the well-known knapsack problem. Hence, this chapter has important applications across various resource allocation problems in computing. To address this "discrete" challenge, we introduce what is, to our knowledge, the first suite of discrete optimistic online learning algorithms, that achieve provable optimality with respect to the regret metric. In the case of the NP-hard knapsack variant, we establish optimality with respect to a *relaxed* regret metric. In this chapter, we build upon the enhanced OFTRL variant introduced in the previous chapter and also develop a new optimistic adap-

tation of Follow the Perturbed Leader (FTPL) (see, e.g., [81]). This new algorithm, Optimistic FTPL (OFTPL), offers improved computational efficiency at the cost of slightly weaker theoretical guarantees. As before, we evaluate our approach on single and bipartite caching networks using standardized datasets and demonstrate its superior performance compared to existing benchmarks.

1.5.3. Optimism Under a Universal Metric

In the previous two chapters, we analyze the performance of our algorithms using the *standard* regret metric, which, as indicated earlier, evaluates an algorithm's decision sequence $\{\boldsymbol{x}_t\}_{t=1}^T$ against the best *fixed* action in hindsight, denoted by $\boldsymbol{x}^\star$. This formulation captures how well an algorithm performs relative to a single, optimal decision. While minimizing this regret is already challenging, real-world systems often require adapting to environments where the optimal decision itself evolves over time. This motivates a shift to the more general and even more challenging *dynamic* regret framework.

Unlike the standard static regret, dynamic regret measures performance against a time-varying sequence of comparators $\{\boldsymbol{x}_t^\star\}_{t=1}^T$, allowing for a non-stationary (changing) benchmark. This setting naturally subsumes static regret as a special case when the comparator remains fixed. More generally, it includes the oracle benchmark, where each $\boldsymbol{x}_t^\star$ is chosen to minimize the corresponding loss function $f_t(\cdot)$ at each time step. As a result, dynamic regret provides a more flexible and fine-grained measure, capturing the inherent variability in real-world decision-making tasks.

This chapter presents the first analysis of the "OFTRL" algorithm, [26], under dynamic comparators. Beyond its intellectual appeal, this analysis yields tighter dynamic regret bounds that are fully modulated by prediction errors, providing a deeper understanding of how optimism interacts with a changing environment. Our results extend the optimistic learning framework beyond static settings, establishing new theoretical guarantees through new customization and analysis of OFTRL based on the concept of *pruning*. We validate the effectiveness of our proposed approach through a series of numerical examples of non-stationary environments.

1.5.4. Optimism in Presence of Memory

Up until now, we assumed that the cost function at each time step t is *oblivious* to past decisions, meaning that the learner's incurred cost depends only on the action taken at t ($f_t(\boldsymbol{x}_t)$). That is, the function remains unchanged regardless of the learner's previous decisions. In this chapter, we extend the framework to a non-oblivious setting, where the cost function at each slot can depend on the *past* m decisions. Specifically, after committing to an action $\boldsymbol{x}_t$, the learner experiences a cost of the form $f_t(\boldsymbol{x}_{t-m}, \ldots, \boldsymbol{x}_{t-1}, \boldsymbol{x}_t)$. This formulation is relevant to many real-world applications. For instance, in caching systems, we might not only care about the performance of the current caching configuration, but also about how different it is from *previous* ones due to a form of switching costs.

Our goal remains, as in the previous chapter, to compete against a benchmark sequence $\{\boldsymbol{u}_t\}_{t=1}^T$. However, the added complexity of memory, or "state", dependence necessitates a more structured analysis. To make the problem tractable, we focus

on settings characterized by a key property: the dependence on the past m decisions is a convex composition of a linear function. This choice is not arbitrary, as it captures several widely studied systems. Most notably, it includes the Linear Quadratic Regulator (LQR) framework (e.g., [82, Ch. 5]), which has been instrumental in applications ranging from control systems to network optimization. Additionally, it encompasses more recent models such as Positive Systems with Linear Costs [83], which have proven useful in a broad range of networking problems [84].

Modeling these stateful systems within the online learning framework was first introduced in the seminal work of [85]. In this chapter, we present the first *optimistic* variant of learning algorithms tailored for this setting. Interestingly, we show that when cost functions exhibit memory effects, optimism necessitates predictions that extend beyond the immediate next cost, requiring forecasts over an *extended horizon*. Our results establish a new connection between the *memory* and *delay* variants of online learning. The latter, delayed online learning, has recently been linked to the optimistic paradigm [86]. Hence, our work takes a first step toward unifying these three fundamental online learning variants. We demonstrate the effectiveness of this *horizon-based* optimism through a series of numerical experiments, highlighting its advantages in state-dependent decision-making problems.

1.6. Technical Progression of the Chapters

Different from the application-driven structure outlined in the previous four subsections, another key perspective for organizing the chapters is through their technical setup and objectives. This perspective is visualized in Fig. 1.2.

At the core of this thesis is the optimistic learning problem, which is parameterized by a set of arbitrary convex time-indexed functions and their corresponding predictions (recall Fig. 1.1). Chapter 2 focuses on minimizing static regret, the difference between our chosen actions, and the best fixed action in hindsight, denoted by $\boldsymbol{x}^\star$. Chapter 3 extends this framework to discrete domains (indicated by the curly braces in the action definition), adopting the notion of α-static regret. Here, the goal is to match a caching configuration $\tilde{\boldsymbol{x}}$ whose cost (i.e., miss ratio) is at most a small multiple $\alpha \geq 1$ of the best possible cost. In some cases, this ratio can be reduced to 1, meaning we recover exactly the performance of the best caching configuration: $\tilde{\boldsymbol{x}} = \boldsymbol{x}^\star$. However, since discrete problems are often NP-hard, directly competing with $\boldsymbol{x}^\star$ is computationally infeasible even with full knowledge of future costs, and we often have to settle with some higher approximation multiple $\alpha > 1$.

In Chapter 4, we shift from static comparators to dynamic ones, competing thus with a *sequence* $\{\boldsymbol{u}_t\}_{t=1}^T$. Here, we return to continuous action spaces (but not necessarily limited to the caching-based box constraints), isolating the challenge of tracking a moving benchmark. Chapter 5 then introduces a further complexity: cost functions with memory. In this setting, the benchmark is no longer a sequence of *instantaneous* optimal decisions but rather a *policy* that accounts for the *history* of actions, leading to a comparison in terms of policy regret. While all chapters present algorithms that provably achieve their respective benchmarks, it is important to note that this simplified overview abstracts away many underlying assumptions. These assumptions will be explicitly stated and formalized in the corresponding chapters.

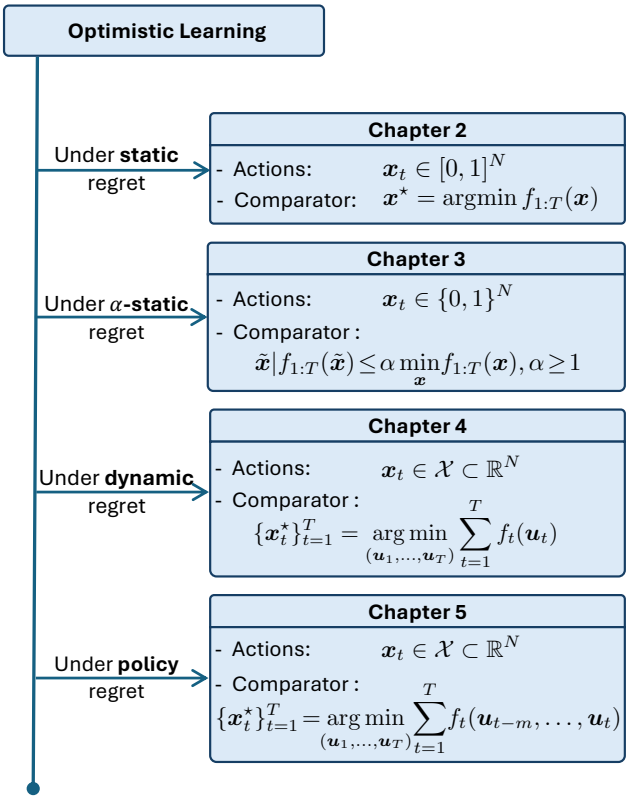


Figure 1.2: Technical progression of the chapters. Each chapter adopts a distinct regret metric suited to its actions set and choice of comparator(s).

2

Optimistic Learning for Continuous Domains: Coded Caching

In this chapter, we take the first step into designing sequential decision-making algorithms that are both robust and prediction-adaptive in the context of the continuous caching algorithm. Specifically, we explore the use of an optimistic learning algorithm that leverages recommendations as predictions for future user requests in coded-caching networks. Since user adherence to recommendations is uncertain, the quality of these predictions remains unknown. We demonstrate that the proposed algorithm achieves a high hit ratio when predictions are accurate while maintaining strong performance guarantees even in the worst-case scenario where all predictions fail (i.e., users ignore the recommendations). This is important in modern caching systems where often the users receive content viewing recommendations from a recommendation system (*RecSys*). For instance, recommendations are a standard feature in streaming platforms such as YouTube and Netflix [60]; but also in social network platforms such as Facebook and Twitter, which moderate the users' viewing feeds [61].

Recommendations as predictions. Not surprisingly, the interplay between recommendations and caching attracted recent attention, and prior works aimed to

The content of this chapter has been published in

- Naram Mhaisen, George Iosifidis, and Douglas Leith. "Online Caching with Optimistic Learning". In: *Proceedings of the International Federation for Information Processing (IFIP) Networking Conference*. 2022.
- Naram Mhaisen, George Iosifidis, and Douglas Leith. "Online Caching with No Regret: Optimistic Learning via Recommendations". In *IEEE Transactions on Mobile Computing* 23.5 (2024), pp. 5949–5965.

increase the caching hits or reduce routing costs, by either recommending already-cached files to users or through the joint optimization of caching and recommendation decisions [67, 68, 69, 87, 88, 89]. These important works, however, consider static caching models and require knowing in advance the users' expected requests and their propensity to follow the recommendations, or make certain assumptions on the structure of the loss function, e.g., strongly-convex.

Changing vantage point, one can observe that since recommendations bias the users towards viewing certain contents, they can effectively serve as predictions of the forthcoming requests. This prediction information, if properly employed, can hugely improve the efficacy of caching policies. Nevertheless, the caching policy needs to adapt to the accuracy of recommendations (i.e., of the predictions) and the users' propensity to follow them – which is typically unknown and potentially time-varying. Otherwise, the caching performance might as well deteriorate by following these misleading *hints* about future requests. The goal of this chapter is to tackle exactly this challenging new problem and answer the question: *Can we leverage untrusted predictions in caching systems?* We answer this question in the affirmative by *proposing online learning-based caching policies that utilize predictions (of unknown quality) to boost performance, if those predictions are accurate, while still maintaining robust performance bounds otherwise.*

Chapter Notation. We use calligraphic capital letters, e.g., $\mathcal{X}$ to denote sets. Vectors are denoted with bold-face small letters[1], e.g., $\boldsymbol{a}$, and we use the subscript t to highlight a vector's dependence on a specific time slot e.g., $\boldsymbol{a}_t$. Scalars are denoted with regular letters and can as well depend on the time, e.g., h_t. When a component of the vector is indexed, the subscript is repurposed to denote that component's (multi-)index, while the t moves to the superscript. i.e., for the d-dimensional vector $\boldsymbol{a}_t$ we write $\boldsymbol{a}_t = (a_1^t, a_2^t, \dots, a_d^t)$. We denote with $\{\boldsymbol{a}_t\}_{t=1}^T$ the sequence of vectors or parameters from slot $t=1$ up to slot T; whenever the horizon is not relevant we use $\{\boldsymbol{a}_t\}_t$. We also use the shorthand sum notation for scalars $b_{1:t} = \sum_{i=1}^T b_i$ and the element-wise sum of vectors $\boldsymbol{a}_{1:t} = \sum_{i=1}^T \boldsymbol{a}_i$. We denote with $[T]$ the integer set $1, 2, \cdots, T$. We make use of the indicator function $I_{\mathcal{X}}(\boldsymbol{x})$, which evaluates to $I_{\mathcal{X}}(\boldsymbol{x})=0$ if $x \in \mathcal{X}$ and ∞ otherwise. The notation is re-stated in more details in Table 2.1.

2.1. Methodology and Contributions

Our approach is based on the theory of Online Convex Optimization (OCO) that was introduced in [90] and has since been applied in several decision problems [22]. The basic premise of OCO is that a learner (here the caching system) selects in each slot t a decision vector $\boldsymbol{x}_t$ from a convex set $\mathcal{X}$, without knowing the t-slot convex performance function $f_t(\boldsymbol{x})$, that changes with time. The learner's goal is to minimize the growth rate of *regret* $R_T = \sum_{t=1}^T f_t(\boldsymbol{x}^\star) - f_t(\boldsymbol{x}_t)$, where $\boldsymbol{x}^\star = \arg\max_{\boldsymbol{x}\in\mathcal{X}} \sum_{t=1}^T f_t(x)$ is the benchmark solution designed with hindsight, i.e., with access to the entire sequence of future functions $\{f_t\}_{t=1}^T$. The online caching problem fits squarely in this setup, where $f_t(\boldsymbol{x})$ depends on the users' requests and is unknown

[1]The only exception is the vector $\boldsymbol{F}_t$, which was done for better readability of section 2.5

when the caching is decided. And previous works [55, 56, 57, 58] have proved that OCO-based caching policies achieve $R_T = \mathcal{O}(\sqrt{T})$, thus ensuring asymptotically zero average regret: $\lim_{T\to\infty} R_T/T = 0$.

Different from these important studies, we extend the learning model to include predictions that are available through the content recommendations. Improving the regret of learning policies via predictions is a relatively new area in machine learning research. For instance, [72] focuses on the *competitive-ratio* metric and developed algorithms that use untrusted predictions while maintaining worst-case performance bounds; while [70] applied similar ideas to the paging problem. However, it was shown in [74] that such competitive-ratio algorithms cannot ensure sublinear regret, which is the performance criterion we employ here, in line with all recent works [55, 56, 57, 58, 59].

For regret-minimization with predictions, [91] used predictions for the function gradient $\nabla f_t(\boldsymbol{x}_t)$ with guaranteed quality to reduce R_T from $\mathcal{O}(\sqrt{T})$ to $\mathcal{O}(\log T)$; and [92] enhanced this result by allowing some predictions to fail the quality condition. A different line of works uses *regularizing functions*, which enable the learner to adapt to the predictions' quality [26, 93]. This idea is more promising for the caching problem, where the recommendations might be inaccurate, or followed by the users for only arbitrary time windows; thus, we used it as a starting point to develop our caching learning frameworks.

In specific, our approach relies on the Follow-The-Regularized-Leader (FTRL) algorithm [94], which we extend with predictions that offer *optimism* by reducing the uncertainty about the next-slot functions. We study different versions of the caching problem. First, we design a policy (OFTRL) for the bipartite caching model [30], which generalizes the standard single cache case [54, 70]. In fact, the bipartite model represents a wide range of caching systems including CDNs, Edge caching and Femtocaching scenarios, D2D caching networks, and so on. Theorem 2.1 proves that R_T is proportional to prediction errors ($\|\boldsymbol{c}_t - \widetilde{\boldsymbol{c}}_t\|^2$), diminishing to zero for perfect predictions; while still meeting the best achievable bound $\mathcal{O}(\sqrt{T})$ for the regular OCO setup (i.e., without using predictions) [58] even if all predictions fail. We continue with the *elastic* caching problem, where the system resizes the used caches at each slot based, e.g., on volatile storage leasing costs [95, 96, 97]. The aim is to maximize the caching utility subject to a time-average budget constraint for the storage capacity costs. This places the problem in the realm of constrained-OCO [98, 99, 100, 101]. Using a new saddle point analysis with predictions, we prove Theorem 2.4, which reveals how the regret and the budget violation depend on the cache sizes and prediction errors, and how one can prioritize one metric over the other while achieving sublinear growth rates for both.

The above algorithms utilize the RecSys as the only source to predict the next time-slot cost function gradient $\widetilde{\boldsymbol{c}}_{t+1}$. In many cases, however, content providers might have access to *multiple* such sources. For example, a statistical user profiling model, a deep learning-based predictive model for content requests [102], or even another RecSys, see [103] and follow-up works. Those sources can be used to obtain multiple, and possibly contradicting, predictions. Our final contribution is, therefore, a *meta-learning* caching framework that utilizes predictions from multi-

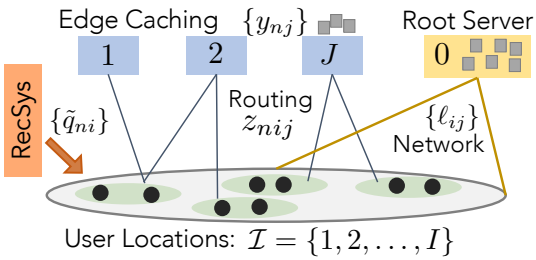


Figure 2.1: **System Model**. A network of J caches serves file requests from a set of I users locations. Unserved requests are routed to the Root Server. Caching decisions are aided via the recommendations provided by the RecSys.

ple sources to achieve the same performance as a caching system that used the best such source to start with. We show that the regret in the case of multiple sources can be strictly negative depending on the request sequence and the existence of a high-accuracy predictor. At the same time, the meta-learning caching framework maintains sublinear regret when all predictors fail. Finally, we show that this framework can also be applied in cases with a single RecSys, and discuss its pros and cons compared to the proposed regularization-based optimistic caching solutions.

In summary, the contributions in this chapter can be grouped as follows:
• Introducing an online learning framework for bipartite and elastic caching networks that leverages predictions to achieve a regret that is upper-bounded by *zero* for perfect recommendations and remains sub-linear $\mathcal{O}(\sqrt{T})$ for arbitrary bad recommendations. The results are based on a new analysis technique that improves the bounds by a factor of $\sqrt{2}$ compared to the state-of-art optimistic-regret bounds [26].

• Introducing a meta-learning framework that utilizes predictions from *multiple* sources and learns which of them to use, if any at all. This framework achieves regret similar to the best predictor-based system, and maintains sub-linear regret if all predictors fail.

• Evaluating the policies using various request models and real datasets [104, 105] and compares them with *(i)* the best in hindsight benchmark; and *(ii)* the online gradient descent policy, which is known to achieve the best possible regret bound (without using predictions) and outperforms other policies [55, 58].

The chapter presents *conceptual innovations*, i.e., using recommendations as an untrusted prediction source for caching decisions, and leveraging different online caching algorithms in an optimistic meta-learning algorithm; as well as *technical contributions* such as the tightened bound of optimistic proximal FTRL (Theorem 2.1) and the new optimistic proximal FTRL algorithm with budget constraints (Theorem 2.4). While we focus on data caching, the proposed algorithms can be directly applied to caching of services and code libraries in edge computing systems.

Table 2.1: Key notation for the chapter.

Parameters	Physical Meaning
Caching Network	
$\mathcal{J}(J)$	Set (number) of caches
$\mathcal{I}(I)$	Set (number) of user locations
C_j	Capacity of cache j. $C_j \leq C, \forall j \in \mathcal{J}$
ℓ_{ij}	Indicator for connectivity of location i to cache j
N	Number of files
q_{ni}^t	Request issued by user i for file n at slot t
w_{nij}	Utility for routing a unit of file n from cache j to i
$f_t(\cdot)$	The utility function at slot t
$\boldsymbol{c}_t$	Gradient of the utility function
$\widetilde{\boldsymbol{c}}_t$	A prediction for $\boldsymbol{c}_t$
s_j^t	Price for unit storage in cache j
P	Number of predictors
Decision Variables	
y_{nj}^t	Portion of file n stored at cache j at slot t
z_{nij}^t	Portion of file n routed from cache j to i at slot t
$\boldsymbol{x}$	A shorthand for the concatenated variables (y, z)
Learning Algorithms	
$r_t(\cdot)$	A strongly convex regularizer function (for $t \geq 1$)
$v_t(\cdot)$	The regularized cost function (negative utility): $-f_t(\cdot) + r_t(\cdot)$
h_t	Prediction error $\|\tilde{c}_t - c_t\|$
σ_t	The change in the aggregated root of prediction error $\sqrt{h_{1:t}} - \sqrt{h_{1:t-1}}$
$\boldsymbol{F}_t$	Experts performance vector at slot t
$\boldsymbol{u}_t$	Weight vector used to combine experts' proposals

2.2. System Model and Problem Statement

2.2.1. Model Preliminaries

Network. The caching network includes a set of edge caches $\mathcal{J} = \{1, 2, \ldots, J\}$ and a root cache indexed with 0, as shown in Fig. 2.1. The file requests emanate from a set of user locations $\mathcal{I} = \{1, 2, \ldots, I\}$. The connectivity between $\mathcal{I}$ and $\mathcal{J}$ is modeled with parameters $\boldsymbol{\ell} = \big(\ell_{ij} \in \{0, 1\} : i \in \mathcal{I}, j \in \mathcal{J}\big)$, where $\ell_{ij} = 1$ if cache j can be reached from location i. We consider the general case where the caches have overlapping coverage; thus, each user can be (potentially) served by one or more edge caches. The root cache is within the range of all users in $\mathcal{I}$. This is a general non-capacitated bipartite model, see [43] for an overview of caching models; and extends the celebrated femtocaching model [30] since the link qualities (which are captured through the utility gains; see below) not only may vary with time, but can do so in an arbitrary (i.e., non-stationary) fashion. This latter feature is particularly

important for the realistic modeling of volatile wireless edge caching systems [106, 107].

Requests. The system operation is time slotted, $t=1,2,\ldots,T$. Users submit requests for obtaining files from a library $\mathcal{N}$ of N files with unit size; we note that the analysis can be readily extended to files with different sizes. This will be made clear in Sec. 2.2.2. Parameter $q_{ni}^t \in \{0,1\}$ indicates the submission of a request for file $n \in \mathcal{N}$ by a user at location $i \in \mathcal{I}$ in the beginning of slot t. At each slot we assume there is one request[2]; i.e., the caching decisions are updated after every request, as in LFU and LRU policies, [108, 109]. Hence, the request process comprises successive vectors $\boldsymbol{q}_t = (q_{ni}^t \in \{0,1\} : n \in \mathcal{N}, i \in \mathcal{I})$ from the set:

$$\mathcal{Q} = \left\{ \boldsymbol{q} \in \{0,1\}^{N \cdot I} \;\middle|\; \sum_{n \in \mathcal{N}} \sum_{i \in \mathcal{I}} q_{ni} = 1 \right\}.$$

We make no assumptions for the request pattern; it might follow a fixed or time-varying distribution that is unknown to the system; and can be even selected strategically by an *adversary* aiming to degrade the caching operation. If a policy's performance is satisfactory under this model, it is ensured to achieve (at least) the same performance for other request models.

Recommendations. There is a recommender system (*RecSys*) that suggests files to each user $i \in \mathcal{I}$, see [60] for the case of Netflix. User i requests one of the recommended files with a certain probability that captures the user's propensity to follow the recommendations. Unlike prior works that consider these probabilities fixed [67, 110], we model them as unknown and possibly time-varying. Namely, no assumption on their quality is guaranteed to remain valid.

A key point in our approach is that the content recommendations, if properly leveraged, can serve as *predictions* for the next-slot requests which are otherwise unknown. We denote with $\widetilde{\boldsymbol{q}}_t$ the prediction for the request $\boldsymbol{q}_t$ that the system will receive at the beginning of slot t, and we assume that $\widetilde{\boldsymbol{q}}_t$ is available at the end of slot $t-1$, i.e., when the RecSys provides its recommendations. Essentially, the recommender system serves as an indirect mechanism for predicting the next request, and this can happen in various ways. For example, the caching system can set $\widetilde{q}_{\hat{n}\hat{i}}^{\,t+1} = 1$ and $\widetilde{q}_{ni}^{\,t+1} = 0, \forall (n,i) \neq (\hat{n},\hat{i})$, where $(\hat{n},\hat{i})$ is the request with the highest predicted probability (top recommended file)[3]. In section 2.5, we study the case where a set $\mathcal{P} = \{1,\ldots,P\}$ of P different predictors (other than the recommendation system) are available, each one offering a prediction $\boldsymbol{q}_t^{(p)}, p \in \mathcal{P}$ at every slot t.

Caching. Each cache $j \in \mathcal{J}$ stores up to $C_j << N$ files, while the root cache stores the entire library, i.e., $C_0 \geq N$. We also define $C = \max_{j \in \mathcal{J}} C_j$. Following the femtocaching model [30], we perform caching using the *Maximum Distance Separable* (MDS) codes, where files are split into a fixed number of F chunks, including redundancy chunks. A user can decode the file if it receives any F-sized subset of its

[2]The proposed policies will still deliver the same regret guarantees when requests are batched before an update. However, the Lipchitz constant will be scaled according to the batch size, and this will affect accordingly the constant factor of the guarantees.

[3]Note that our caching policy is orthogonal to the mechanism that maps the recommendations to predictions. Namely, our results will be stated in terms of the prediction error.

chunks. For large values of F, the MDS model allows us to use continuous caching variables.[4] Hence, we define the variable $y_{nj}^t \in [0,1]$ which denotes the portion of F chunks of file $n \in \mathcal{N}$ stored at cache $j \in \mathcal{J}$, and we introduce the t-slot caching vector $\boldsymbol{y}_t = (y_{nj}^t : n \in \mathcal{N}, j \in \mathcal{J})$ that belongs to set:

$$\mathcal{Y} = \left\{ \boldsymbol{y} \in [0,1]^{N \cdot J} \;\middle|\; \sum_{n \in \mathcal{N}} y_{nj} \leq C_j,\ j \in \mathcal{J} \right\}.$$

We note that our model can be readily extended to files of different size, by replacing the capacity constraint in $\mathcal{Y}$ with

$$\sum_{n \in \mathcal{N}} v_n y_{nj} \leq C_j,\ j \in \mathcal{J}$$

For some general size vector $v \in \mathbb{R}_+^N$. Such a change will not affect the mathematical characteristics of the optimization problem (both sets correspond to linear constraints).

Routing. Since each user location $i \in \mathcal{I}$ may be connected to multiple caches, we need to introduce routing variables. Let z_{nij}^t denote the portion of request q_{ni}^t served by cache j. In the MDS caching model the requests can be simultaneously served from multiple caches and, naturally, we restrict[5] the amount of chunks not to exceed F. Hence, the t-slot routing vector $\boldsymbol{z}_t = (z_{nij}^t \in [0,1] : n \in \mathcal{N}, i \in \mathcal{I}, j \in \mathcal{J})$ is drawn from:

$$\mathcal{Z} = \left\{ \boldsymbol{z} \in [0,1]^{N \cdot J \cdot I} \;\middle|\; \sum_{j \in \mathcal{J}} z_{nij} \leq 1,\ n \in \mathcal{N}, i \in \mathcal{I} \right\}.$$

Requests that are not (fully) served by the edge caches $\mathcal{J}$ are served by the root server that provides the missing chunks. This decision needs not to be explicitly modeled as it is directly determined by the routing vector $\boldsymbol{z}_t$.

2.2.2. Problem Statement

Cache Utility & Predictions. We use parameters $w_{nij} \in [0, w]$ to model the system utility when delivering a chunk of file $n \in \mathcal{N}$ to location $i \in \mathcal{I}$ from cache $j \in \mathcal{J}$, instead of using the root server. This general utility model offers the ability to capture bandwidth savings or delay reductions, as well as other common objectives of edge-caching, in both wired and wireless networks. It is noteworthy that under this model, the caching benefits may vary for each cache and user location, and they can also change over time. Additionally, the problem of maximizing cache hits can be viewed as a specific instance within this context (when $w_{nij} = 1$), as discussed

[4] Large files are composed of thousands of chunks, leading to a small chunk size (compared to the original file). This induces practically negligible errors in the utility function [43, Sec. 3.3]. In addition, even for exact discrete caching, relaxing the integrality constraints and solving the continuous version is an essential first step which is then followed by a randomized rounding technique (see, e.g., [64, Sec. 6]).

[5] This practical constraint is called the *inelastic* model and compounds the problem, cf. [58] for the simpler elastic model.

in [32]. To streamline presentation we introduce vector $\boldsymbol{x}_t = (\boldsymbol{y}_t, \boldsymbol{z}_t) \in \mathbb{R}^m$, with $m = NIJ + NJ$, and define the system utility in slot t as:

$$f_t(\boldsymbol{x}_t) = \sum_{n\in\mathcal{N}} \sum_{i\in\mathcal{I}} \sum_{j\in\mathcal{J}} w_{nij} q_{ni}^t z_{nij}^t \ ,$$

and we denote its gradient $\boldsymbol{c}_t = \nabla f_t(x_t)$. As it will become clear, our analysis holds also for non-linear concave functions $f_t(x)$; this generalization is useful in case, e.g., we wish to enforce fairness in the dispersion of caching gains across user locations [96].

The main challenge in online caching is the following: at the end of each slot t where we need to decide the cache configuration, the utility function f_{t+1} is not available. Indeed, this function depends on the next-slot request $\boldsymbol{q}_{t+1}$, which is made known only after $\boldsymbol{y}_{t+1}$ has been decided and fixed[6], see [55, 58, 70]. Note that this is also the operation timing of the LRU/LFU policies [108, 109]. However, the recommendations provided to users can be used to form the vector of predicted requests $\widetilde{\boldsymbol{q}}_{t+1}$, which in turn can be used to create a prediction for the gradient of the next slot function $\widetilde{f}_{t+1}(\cdot)$. This predicted gradient, denoted $\widetilde{\boldsymbol{c}}_{t+1}$, suffices for designing the proposed optimistic algorithms.

Benchmark. In such learning problems, it is important to understand the objective that our algorithm aims to achieve. If we had access to an oracle for all requests over a horizon of T slots $\{\boldsymbol{q}_t\}_{t=1}^T$ (and the utility parameters) we could have devised the utility-maximizing static caching and routing policy $\boldsymbol{x}^\star = (\boldsymbol{y}^\star, \boldsymbol{z}^\star)$, by solving the following convex optimization problem:

$$\begin{aligned} \mathbb{P}_1: \quad \max_{\boldsymbol{x}} . \quad & \sum_{t=1}^{T} f_t(\boldsymbol{x}) \\ \text{s.t.} \quad & z_{nij} \leq y_{nj}\ell_{ij}, \quad i \in \mathcal{I}, j \in \mathcal{J}, n \in \mathcal{N}, \\ & \boldsymbol{z} \in \mathcal{Z}, \quad \boldsymbol{y} \in \mathcal{Y}, \end{aligned} \tag{2.1}$$

where (2.1) ensure that the routing decisions for each requested file use only caches that store enough chunks of that file. Let us define the convex set of constraints:

$$\mathcal{X} = \big\{ \{\mathcal{Y} \times \mathcal{Z}\} \cap \{(2.1)\} \big\} \tag{2.2}$$

that we will use henceforth to streamline presentation.

Clearly, this hypothetical solution $\boldsymbol{x}^\star$ can be designed only with *hindsight* and is the benchmark for evaluating our online learning policy π which outputs $\{\boldsymbol{x}_t\}_t$. Thus, in line with prior works, we use the metric of static regret:

$$R_T(\pi) = \sup_{\{f_t\}_{t=1}^T} \left[\sum_{t=1}^{T} f_t(\boldsymbol{x}^\star) - \sum_{t=1}^{T} f_t(\boldsymbol{x}_t) \right], \tag{2.3}$$

[6]In our case, since the routing is directly shaped by the caching, this restriction affects also z_{t+1}.

which quantifies the performance gap of π from $\boldsymbol{x}^\star$, for any possible sequence of requests or, equivalently, functions $\{f_t\}_t$. Our goal is to find a policy that achieves sublinear regret, $R_T(\pi) = \mathcal{O}(T)$, thus ensuring the average performance gap R_T/T will diminish as T grows. This policy, similar to other online policies, decides $\boldsymbol{x}_{t+1}$ at the end of each slot t using the previous utility functions $\{f_\tau\}_{\tau=1}^t$ and the next-slot prediction $\tilde{f}_{t+1}$ devised from the RecSys.

Note that, in principle, the regret metric can be negative. This is especially true for optimistic policies. To see why, recall that $\boldsymbol{x}^\star$ is the best *fixed* caching configuration, whereas $\boldsymbol{x}_t$ is allowed to change for each t. Hence, it might happen, e.g., that $\boldsymbol{x}_t$ performs better than $\boldsymbol{x}^\star$ on some steps, $f_t(\boldsymbol{x}_t) \geq f_t(\boldsymbol{x}^\star)$, while performing similar to $\boldsymbol{x}^\star$ in the remaining ones $f_t(\boldsymbol{x}_t) \approx f_t(\boldsymbol{x}^\star)$. Of course, the occurrence of such an event depends on the request sequence and the policy that determines $\boldsymbol{x}_t$.

On the other hand, there are stricter benchmarks such as those that allow the hindsight policy to pick a different decision in each slot, $\boldsymbol{x}_t^\star \in \arg\max_{\boldsymbol{x}\in\mathcal{X}} f_t(\boldsymbol{x})$. These benchmarks give rise to the dynamic regret metric[7], which compares the learning algorithm with $\sum_{t=1}^T f_t(\boldsymbol{x}_t^\star)$. However, it is well established that achieving sublinear dynamic regret requires additional assumptions on the variability of the function $f_t(\cdot)$ (see, e.g., the discussion in the introduction section of [111]). In addition, for the caching problem in particular, such a dynamic policy, (i.e., $\{\boldsymbol{x}_t^\star\}_{t=1}^T$), will alter the cache state *at every time-step*, inducing prohibitive switching cost. This is in contrast to the presented policies that *converge* to the best fixed solution. In the following sections, we show that optimism can greatly decrease the upper bound on R_T, increasing the chances of negative regret.

2.3. Optimistic Bipartite Caching

Unlike recent caching solutions that rely on Online Gradient Descent (OGD) [55] or on the Follow-the-Perturbed-Leader (FTPL) policy [58], our approach draws from the *Follow-The-Regularized-Leader* (FTRL) policy, cf. [112], appended with prediction-adaptive regularizers. A key element in our proposal is the *optimism* emanating from the availability of predictions, namely the content recommendations that are offered to users by the RecSys in each slot.

Let us begin by defining the proximal regularizers[8]:

$$r_0(\boldsymbol{x}) = I_{\mathcal{X}}(\boldsymbol{x}), \quad r_t(\boldsymbol{x}) = \frac{\sigma_t}{2}\|\boldsymbol{x} - \boldsymbol{x}_t\|^2, \ t \geq 1 \tag{2.4}$$

where $\|\cdot\|$ is the Euclidean norm, and recall that $I_{\mathcal{X}}(\boldsymbol{x}) = 0$ if $x \in \mathcal{X}$ and ∞ otherwise. We apply properly selected regularizing parameters, which change the strong convexity of r_t according to the predictions' quality until t, and also ensure $r_t(\boldsymbol{x}) \geq 0, \forall t$, namely:

$$\sigma_1 = \sigma\sqrt{h_1}, \quad \sigma_t = \sigma\left(\sqrt{h_{1:t}} - \sqrt{h_{1:t-1}}\right), \quad t \geq 2 \tag{2.5}$$

[7] We refer the reader to [22, Ch. 10] for variations on the regret metric

[8] A proximal regularizer induces a proximal mapping for the objective function; see [113, Ch. 6.1] for the formal definition.

Algorithm 1: Optimistic Bipartite Caching (π_{obc})

Input: $\{\ell_{ij}\}_{(i,j)}$; $\{C_j\}_j$; $\mathcal{N}$; $\boldsymbol{x}_1 \in \mathcal{X}$; $\sigma = \sqrt{2}/D_{\mathcal{X}}$.
Output: $\boldsymbol{x}_t = (\boldsymbol{y}_t, \boldsymbol{z}_t)$, $\forall t$.
for $t = 1, 2, \ldots$ **do**
  Route request q_t according to configuration $\boldsymbol{x}_t$
  Observe system utility $f_t(\boldsymbol{x}_t)$
  Observe the new prediction $\widetilde{c}_{t+1}$
  Update the regularizer $r_{0:t}(\boldsymbol{x})$ using (2.4)-(2.5)
  Calculate the new policy $\boldsymbol{x}_{t+1}$ using (2.6)
end

$$\text{with} \quad h_t = \|\boldsymbol{c}_t - \widetilde{\boldsymbol{c}}_t\|^2,$$

where $\sigma \geq 0$, $\boldsymbol{c}_t = \nabla f_t(\boldsymbol{x}_t)$, and we used the shorthand sum notation $h_{1:t} = \sum_{i=1}^{t} h_i$ for the aggregate prediction errors during the first t slots. The basic step of the algorithm is:

$$\boldsymbol{x}_{t+1} = \arg\min_{\boldsymbol{x} \in \mathbb{R}^m} \Big\{ r_{0:t}(\boldsymbol{x}) - (\boldsymbol{c}_{1:t} + \langle \widetilde{\boldsymbol{c}}_{t+1}, \boldsymbol{x} \rangle \Big\}, \tag{2.6}$$

which calculates the decision vector using past utility observations $\boldsymbol{c}_{1:t}$, the aggregate regularizer $r_{0:t}(\boldsymbol{x})$ and the prediction $\widetilde{\boldsymbol{c}}_{\boldsymbol{t+1}}$. The update employs the negative gradients as it concerns a maximization problem. Henceforth, we refer to (2.6) as the *optimistic* FTRL (OFTRL) update.

To provide some intuition on (9), ignore for a moment the regularization term. Our decision vector for $t+1$ is then simply the minimization of a linear cost (or, equivalently, the maximization of linear utility) that consists of the total cost witnessed until time t, plus the cost prediction for $t+1$. This is indeed a reasonable objective, provided that the prediction is accurate. However, the solution to any linear program is on the extremes of the decision set. Hence, the adversary can exploit this and make the learner jump between extreme points while placing maximum costs at those (e.g., via wrong predictions). Here comes the role of the regularization term, which stabilizes our decisions since the solution to the quadratic program is no longer on the extreme points. In fact, we want such regularization to be proportional to the witnessed accuracies of the predictions (hence the choices in (8)). This way, if the predictions are accurate, the linear program provides good decisions. Otherwise, the regularization induces a stable quadratic program whose solution cannot be arbitrarily harmed.

Policy π_{obc} is outlined in Algorithm 1. In each iteration, OBC solves a convex optimization problem, (2.6), involving a projection on the feasible set $\mathcal{X}$ (via $r_0(\boldsymbol{x})$). For the latter, one can rely on fast-projection algorithms specialized for caching, e.g., see [55]; while it is possible to obtain a closed-form solution for the OFTRL update for linear functions. We quantify next the performance of Algorithm 1.

Theorem 2.1. *Algorithm 1 ensures the regret bound:*

$$R_T \leq 2\sqrt{1+JC}\sqrt{\sum_{t=1}^{T}\|\boldsymbol{c}_t-\widetilde{\boldsymbol{c}}_t\|^2}.$$

For the proof, we modify the "strong FTRL lemma" [112, Lemma 5] by adding predictions for next-slot utility function. The following lemma bounds the regret in terms of the difference between two values of a strongly convex function evaluated at $\boldsymbol{x}_t$ and at $\boldsymbol{x}_{t+1}$, for each t.

Lemma 2.2. *(Optimistic Strong FTRL Lemma) Let $v_t(\boldsymbol{x}) = -\langle \boldsymbol{c}_t, \boldsymbol{x}_t\rangle + r_t(\boldsymbol{x}_t)$, and $v_{0:t}(\boldsymbol{x}) = -\langle \boldsymbol{c}_{1:t}, \boldsymbol{x}_t\rangle + r_{0:t}(\boldsymbol{x}_t)$. Let $\boldsymbol{x}_{t+1}$ be selected according to (2.6). Then:*

$$R_T \;\leq\; r_{0:T}(\boldsymbol{x}^\star) + \sum_{t=1}^{T} v_{0:t}(\boldsymbol{x}_t) - v_{0:t}(\boldsymbol{x}_{t+1}) - r_t(\boldsymbol{x}_t) + \langle \widetilde{\boldsymbol{c}}_{T+1}, \boldsymbol{x}_{T+1} - \boldsymbol{x}^\star\rangle \quad (2.7)$$

Proof of Lemma 2.2.

$$\begin{aligned}
&\sum_{t=1}^{T} v_t(\boldsymbol{x}_t) - (v_{0:T}(\boldsymbol{x}^\star) - \langle \widetilde{\boldsymbol{c}}_{T+1}, \boldsymbol{x}^\star\rangle)\\
&= \sum_{t=1}^{T} (v_{0:t}(\boldsymbol{x}_t) - v_{0:t-1}(\boldsymbol{x}_t)) - (v_{0:T}(\boldsymbol{x}^\star) - \langle \widetilde{\boldsymbol{c}}_{T+1}, \boldsymbol{x}^\star\rangle)\\
&\leq \sum_{t=1}^{T} v_{0:t}(\boldsymbol{x}_t) + \sum_{t=1}^{T} (-v_{0:t-1}(\boldsymbol{x}_t)) - (v_{0:T}(\boldsymbol{x}_{T+1}) - \langle \widetilde{\boldsymbol{c}}_{T+1}, \boldsymbol{x}_{T+1}\rangle) \text{ (by def. of } \boldsymbol{x}_{T+1})\\
&\overset{(a)}{=} \sum_{t=1}^{T} v_{0:t}(\boldsymbol{x}_t) - v_0(\boldsymbol{x}_1) + \sum_{t=1}^{T-1} (-v_{0:t}(\boldsymbol{x}_{t+1})) - v_{0:T}(\boldsymbol{x}_{T+1}) + \langle \widetilde{\boldsymbol{c}}_{T+1}, \boldsymbol{x}_{T+1}\rangle\\
&\overset{(b)}{\leq} \sum_{t=1}^{T} \big(v_{0:t}(\boldsymbol{x}_t) - v_{0:t}(\boldsymbol{x}_{t+1})\big) + \langle \widetilde{\boldsymbol{c}}_{T+1}, \boldsymbol{x}_{T+1}\rangle
\end{aligned}$$

where equality (a) follows by reindexing the second sum (i.e., changing the sum index from t to $t+1$), and inequality (b) by dropping the non-positive term $-v_0(\boldsymbol{x}_1) = -r_0(\boldsymbol{x}_1)$ and appending the term $-v_{0:T}(\boldsymbol{x}_{T+1})$ to the second sum. Thus, we have:

$$\sum_{t=1}^{T} v_t(\boldsymbol{x}_t) - v_{0:T}(\boldsymbol{x}^\star) + \langle \widetilde{\boldsymbol{c}}_{T+1}, \boldsymbol{x}^\star\rangle \leq \sum_{t=1}^{T} \big(v_{0:t}(\boldsymbol{x}_t) - v_{0:t}(\boldsymbol{x}_{t+1})\big) + \langle \widetilde{\boldsymbol{c}}_{T+1}, \boldsymbol{x}_{T+1}\rangle.$$

Expanding the definition of $v_t(\boldsymbol{x}_t)$ and rearranging give the regret inequality:

$$\begin{aligned}
\langle \boldsymbol{c}_{0:T}, \boldsymbol{x}^\star\rangle - \sum_{t=1}^{T}\langle \boldsymbol{c}_t, \boldsymbol{x}_t\rangle \leq r_{0:T}(\boldsymbol{x}^\star) + \sum_{t=1}^{T} \big(v_{0:t}(\boldsymbol{x}_t) - v_{0:t}(\boldsymbol{x}_{t+1})\big)\\
- r_t(\boldsymbol{x}_t) + \langle \widetilde{\boldsymbol{c}}_{T+1}, (\boldsymbol{x}_{T+1} - \boldsymbol{x}^\star)\rangle.
\end{aligned}$$

Noticing that the LHS is essentially the regret defined in (2.3) for $f_t(\boldsymbol{x}) = \langle \boldsymbol{c}_t, \boldsymbol{x}\rangle$

completes the proof [9]. □

With the weak assumption that the caching system will be notified upon the serving of the last request, we can set $\widetilde{\boldsymbol{c}}_{T+1} = 0$ and hence cancel the last term in the above inequality. Otherwise, it will be an additional constant factor in the regret bound[10]. Next, we will make use of the following results to bound each $v_{0:t}(\boldsymbol{x}_t) - v_{0:t}(\boldsymbol{x}_{t+1})$ term:

Lemma 2.3. *[112, Lemma 7] let $\phi_1 : \mathbb{R}^n \to \mathbb{R}$ be a convex function such that $\boldsymbol{x}_1 = \arg\min_{\boldsymbol{x}} \phi_1$. Let ψ be a convex function such that $\phi_2(\boldsymbol{x}) = \phi_1(\boldsymbol{x}) + \psi(\boldsymbol{x})$ is strongly convex w.r.t norm $\|\cdot\|$. Then, for any $\boldsymbol{b} \in \partial\psi(\boldsymbol{x}_1)$ and $\boldsymbol{x}'$, we have that $\phi_2(\boldsymbol{x}_1) - \phi_2(\boldsymbol{x}') \leq \frac{1}{2}\|\boldsymbol{b}\|_\star^2$.*

Now we are ready to prove Theorem 1:

Proof of Theorem 1. We start by applying Lemma 2.3 to the result in (2.7). Namely, we select:

$$\phi_1(\boldsymbol{x}) = v_{0:t}(\boldsymbol{x}) + \langle \boldsymbol{c}_t, \boldsymbol{x}_t\rangle - \langle \widetilde{\boldsymbol{c}}_t, \boldsymbol{x}_t\rangle, \text{ and}$$
$$\phi_2(\boldsymbol{x}) = \phi_1(\boldsymbol{x}) - \langle \boldsymbol{c}_t, \boldsymbol{x}_t\rangle + \langle \widetilde{\boldsymbol{c}}_t, \boldsymbol{x}_t\rangle.$$

This way, we have that $\boldsymbol{x}_t = \arg\min \phi_1(\boldsymbol{x}_t)$, $\phi_2(\boldsymbol{x}) = v_{0:t}(\boldsymbol{x})$, $\psi(\boldsymbol{x}) = \langle -\boldsymbol{c}_t + \widetilde{\boldsymbol{c}}_t, \boldsymbol{x}\rangle$, and $(-\boldsymbol{c}_t + \widetilde{\boldsymbol{c}}_t) \in \partial\psi(\boldsymbol{x})$.

Then, dropping the non-positive terms $-r_t(\cdot)$ in (2.7), setting $\widetilde{\boldsymbol{c}}_{T+1} = 0$, and defining the norm $\|\cdot\|_{(t)} = \sqrt{\sigma_{1:t}}\|\cdot\|$ so that the regularizer $r_{1:t}(\boldsymbol{x})$ is 1-strongly-convex w.r.t. $\|\cdot\|_{(t)}$, we get:

$$R_T \leq r_{1:T}(\boldsymbol{x}^\star) + \frac{1}{2}\sum_{t=1}^{T} \|\boldsymbol{c}_t - \widetilde{\boldsymbol{c}}_t\|_{(t),\star}^2, \quad \forall \boldsymbol{x}^\star \in \mathcal{X}. \tag{2.8}$$

Now, we have that $r_t \leq \frac{\sigma_t}{2} D_{\mathcal{X}}^2$, where $D_{\mathcal{X}}$ is the Euclidean diameter of the set $\mathcal{X}$ i.e., $\forall \boldsymbol{x}, \boldsymbol{x}_t \in \mathcal{X}$:

$$\|\boldsymbol{x} - \boldsymbol{x}_t\|^2 = \sum_{n,j}(y_{nj} - y_{nj}^t)^2 + \sum_{n,i,j}(z_{nij} - z_{nij}^t)^2 \overset{(a)}{\leq} \sum_{n,j}|y_{nj} - y_{nj}^t| + \sum_{n,i,j}|z_{nij} - z_{nij}^t|$$
$$\overset{(b)}{\leq} 2(JC + 1) \triangleq D_{\mathcal{X}}^2$$

where (a) holds as $y_{nj}, z_{nij} \in [0,1], \forall n, i, j$; (b) holds by the triangle inequality and definitions of $\mathcal{Y}$, $C \triangleq \max_j C_j$, and the fact that the routing variables differ at one coordinate only. To see why, recall that we serve one request per time slot and we set the routing variable $\boldsymbol{z}_t$ after observing that request $\boldsymbol{q}_t$. Hence, we can modify the routing variables and set $z_{nij}^\star = z_{nij}^t = 0\ \forall n \neq n', i \neq i', j \neq j'$, where

[9] This inequality holds for all $\boldsymbol{c}_t$, including $\sup_{\boldsymbol{c}}\langle \boldsymbol{c}, \boldsymbol{x}\rangle$.

[10] It is possible to slightly change the semantics of the algorithm to avoid this rare case by making the adversary first commit and hide the cost function, and then the learner pick the action, see [28, Thm 7.29].

$q_{n'i'} = 1 \wedge \ell_{i'j'=1}$. Due to the structure of $f_t(\cdot)$, the utility of these modified $z^\star_{nij}$ and z^t_{nij} will not change (compared to their utility before modification). In words, knowing the requested file n', and the location from which it is requested i', there will be no utility from routing a non-requested file $n \neq n'$, or routing from a non-connected cache $j \neq j'$. Thus, we can zero these variables and get a smaller value for $D_{\mathcal{X}}$, without affecting the utility values.

Using this diameter bound, and the fact that the dual norm of $\|\boldsymbol{x}\|_{(t)}$ is $\|\boldsymbol{x}\|_{(t),\star} = \|\boldsymbol{x}\|/\sqrt{\sigma_{1:t}}$, inequality (2.8) can be written as:

$$R_T \leq \frac{\sigma_{1:T}}{2} D^2_{\mathcal{X}} + \frac{1}{2}\sum_{t=1}^{T} \frac{h_t}{\sigma_{1:t}}. \tag{2.9}$$

Note that the sum $\sigma_{1:t}$ telescopes and evaluates to $\sigma\sqrt{h_{1:t}}$. Using this observation and substituting it in (2.9), and combining it with [114, Lem. 3.5] to bound the second term as follows $\sum_t^T h_t/\sqrt{h_{1:t}} \leq 2\sqrt{h_{1:T}}$, we eventually get:

$$R_T \leq \frac{\sigma}{2}\sqrt{h_{1:T}} D^2_{\mathcal{X}} + \frac{1}{\sigma}\sqrt{h_{1:T}} \overset{(a)}{\leq} \sqrt{2} D_{\mathcal{X}} \sqrt{h_{1:T}},$$

where (a) is obtained by setting $\sigma = \sqrt{2}/D_{\mathcal{X}}$. Finally, substituting the actual diameter value, namely $D_{\mathcal{X}} = \sqrt{2(JC+1)}$, completes the proof. □

Discussion. Theorem (2.1) shows that the regret does not depend on the library size N and is also modulated by the quality of the predictions; accurate predictions tighten the bound, and in the case of perfect predictions, i.e., when users follow the recommendations, we get negative regret $R_T \leq 0, \forall T$, which is much stronger than the sub-linear growth rates in other works [55, 115]. In fact, even when predictions fail for a constant number of time slots $L \in [T]$, the regret will be constant of the same order $R_T \leq \mathcal{O}(L)$, which is still a significant improvement over any time-dependent bound.

On the other hand, for worst-case prediction, we can still use the bound $\|\boldsymbol{c}_t - \widetilde{\boldsymbol{c}}_t\|^2 \leq 2w^2$, and get:

$$R_T \leq 2\sqrt{2} w \sqrt{JC+1}\sqrt{T} = \mathcal{O}(\sqrt{T})$$

i.e., the regret is at most a constant factor worse than the regret of those policies that do not incorporate predictions[11]. Thus, OBC offers an efficient and safe approach for incorporating predictions in cases where we are uncertain about their accuracy, e.g., either due to the quality of the RecSys or the behavior of users.

Another key point is that the *utility parameters might vary with time* as well. Indeed, replacing $\boldsymbol{w}_t = (w^t_{nij} \leq w, n \in \mathcal{N}, i \in \mathcal{I}, j \in \mathcal{J})$ in $f_t(\boldsymbol{x}_t)$ does not affect the analysis nor the bound. This is important when the caching system employs a wireless network where the link capacities vary, or when the caching utility changes; and we note that this utility can be even *file-specific*. Parameters $\boldsymbol{w}_t$ can be also unknown

[11] The factor is $\sqrt{2}$ compared to the "any-time" version of the bound that does not use predictions, and 2 compared to those that assume a known T.

Algorithm 2: Optimistic Elastic Caching (π_{oec})

Input: $\{\ell_{ij}\}_{(i,j)}$, $\{C_j\}_j$, $\mathcal{N}$, $\lambda_1 = 0$, $x_1 \in \mathcal{X}_e$.
Output: $x_t = (y_t, z_t)$, $\forall t$.
for $t = 1, 2, \ldots$ **do**
  Route request q_t according to configuration x_t
  Observe system utility $f_t(\boldsymbol{x}_t)$ and cost $g_t(\boldsymbol{x}_t)$
  Update the budget parameter λ_{t+1} using (2.11)
  Update the regularizers $r_{0:t}(x)$ using (2.4)-(2.5), and $a_t = at^{-\beta}$
  Observe prediction $\widetilde{c}_{t+1}$ and price s_{t+1}
  Calculate the new policy x_{t+1} using (2.12)
end

when $\boldsymbol{x}_t$ is decided, exactly as it is with $\boldsymbol{q}_t$, and they can be predicted using e.g., channel measurements. Essentially the proposed model drops several restricted assumptions of prior works regarding not only the knowledge of request rates/densities but also about the system state, link quality, and user utilities. Finally, we observe that in case the algorithm is used to optimize the operation of an edge computing system, these parameters can capture the potentially time-varying utility of each computation, for each service (n) and each user-cache pair.

On a technical note, Lemma 2.2 presents a novel theoretical result and enables the improvement of the best known proximal OFTRL bound of [26] by a constant factor of $\sqrt{2}$. This also opens the door for leveraging the modular analysis tools developed in [112] in the optimistic OCO framework for different special cases of the utility functions. This technical result is of independent interest.

Lastly, in the case of more than one request per time slot (e.g., B requests), the Euclidean norm would instead be $D_{\mathcal{X}}^2 = 2(JC + B)$. Therefore, the same results hold in terms of the regret being always sub-linear and commensurate with the prediction accuracy. However, the worst case bounds will of course be scaled by a factor of $\sqrt{B}$ since now we would use $\|\boldsymbol{c}_t - \widetilde{\boldsymbol{c}}_t\|^2 \leq 2B\ w^2$.

2.4. Optimistic Caching in Elastic Networks

We extend our analysis to *elastic* caching networks where the caches can be resized dynamically. Such architectures are important for two reasons. Firstly, there is a growing number of small-size content providers that implement their services by leasing storage on demand from infrastructure providers [116]; and secondly, CDNs often resize their caches responding to the time-varying user needs and operating expenditures [117].

We introduce the t-slot price vector $\boldsymbol{s}_t = (s_j^t \leq s, j \in \mathcal{J})$, where s_j^t is the leasing price per unit of storage at cache j in slot t, and s its maximum value. In the general case, these prices may change arbitrarily over time, e.g., because the provider has a dynamic pricing scheme or the electricity cost changes [95, 96]; hence the caching system has access only to $\boldsymbol{s}_t$ at each slot t. We denote with B_T the budget the system intends to spend during a period of T slots for leasing cache capacity. The

objective is to maximize the caching gains while satisfying:

$$\sum_{t=1}^{T} g_t(\boldsymbol{x}_t) = \sum_{t=1}^{T} \sum_{j\in\mathcal{J}} \sum_{n\in\mathcal{N}} s_j^t y_{nj}^t - B_T \leq 0.$$

In particular, the new benchmark problem in this case is:

$$\mathbb{P}_2: \quad \max_{x\in\mathcal{X}} \sum_{t=1}^{T} f_t(\boldsymbol{x}), \quad \text{s.t.} \sum_{t=1}^{T} g_t(\boldsymbol{x}) \leq 0,$$

which differs from $\mathbb{P}_1$ due to the leasing constraint.

Indeed, in this case the regret is defined as:

$$R_T^{(e)}(\pi) = \sup_{\{f_t\}_{t=1}^T} \left[\sum_{t=1}^{T} f_t(\boldsymbol{x}^\star) - \sum_{t=1}^{T} f_t(\boldsymbol{x}_t) \right],$$

where

$$\boldsymbol{x}^\star \in \mathcal{X}_e \triangleq \{\boldsymbol{x} \in \mathcal{X} \mid (2.1), g_t(\boldsymbol{x}) \leq 0, \forall t\},$$

i.e., $\boldsymbol{x}^\star$ is a feasible point of $\mathbb{P}_1$ with the newly introduced additional restriction to satisfy the budget constraint $g_t(\boldsymbol{x}) \leq 0$ in every slot. In the definition of $\mathcal{X}$, C now denotes the maximum leasable space. Learning problems with time-varying constraints are hard, see impossibility result in [118], and hence require such additional restrictions on the selected benchmarks. We refer the reader to [98] for a related discussion, and to [99, 100] for different benchmarks. Finally, apart from $R_T^{(e)}$, we need also to ensure a sublinear growth rate for the budget violation:

$$V_T^{(e)} = \sum_{t=1}^{T} [g_t(\boldsymbol{x}_t)]_+ .$$

To tackle this new problem we follow a saddle point analysis, which is new in the context of OFTRL.

Namely, we first define a Lagrangian-type function by relaxing the budget constraint and introducing the dual variable $\lambda \geq 0$:

$$\mathcal{L}_t(\boldsymbol{x}, \lambda) = \frac{\sigma_t}{2}\|\boldsymbol{x} - \boldsymbol{x}_t\|^2 - f_t(\boldsymbol{x}) + \lambda g_t(\boldsymbol{x}) - \frac{\lambda^2}{a_t}. \tag{2.10}$$

The last term is a non-proximal regularizer for the dual variable; and we use $a_t = at^{-\beta}$, where parameter $\beta \in [0, 1)$ can be used to prioritize either $R_T^{(e)}$ or $V_T^{(e)}$. The main ingredients of policy π_{oec} are the saddle-point iterations:

$$\lambda_{t+1} = \arg\max_{\lambda\geq 0} \left\{ -\frac{\lambda^2}{a_{t+1}} + \lambda \sum_{i=1}^{t} g_i(\boldsymbol{x}_i) \right\}, \tag{2.11}$$

$$\boldsymbol{x}_{t+1} = \arg\min_{\boldsymbol{x}\in\mathbb{R}^m} \left\{ r_{0:t}(\boldsymbol{x}) + \big(\sum_{i=1}^{t+1} \lambda_i \boldsymbol{s}_i - \boldsymbol{c}_{1:t} - \langle \widetilde{\boldsymbol{c}}_{t+1}\big), \boldsymbol{x}\rangle \right\}, \tag{2.12}$$

and its implementation is outlined in Algorithm 2. Note that we use the same regularizer as in Sec. 2.3 for the primal variables $\boldsymbol{x}_t$, while λ_t modulates the caching decisions by serving as a *shadow price* for the average budget expenditure.

The performance of Algorithm OEC is characterized in the next theorem.

Theorem 2.4. *Algorithm 2 ensures the bounds:*

$$R_T^{(e)} \leq \sqrt{2} D_{\mathcal{X}} \sqrt{\sum_{t=1}^{T} \|\boldsymbol{c}_t - \widetilde{\boldsymbol{c}}_t\|^2} + \frac{aM}{2} T^{1-\beta},$$

$$V_T^{(e)} \leq \sqrt{\frac{2\sqrt{2} D_{\mathcal{X}} T^{\beta}}{a} \sqrt{\sum_{t=1}^{T} \|\boldsymbol{c}_t - \widetilde{\boldsymbol{c}}_t\|^2} + MT - \frac{2R_T^{(e)} T^{\beta}}{a}},$$

where we have used the grouped constants $M = \frac{(sJC)^2}{1-\beta}$.

Proof. Observe that the update in (2.12) is similar to (2.6) but applied to the Lagrangian in (2.10) instead of just the utility, and the known prices when $\boldsymbol{x}_{t+1}$ is decided represent perfect prediction for $g_t(\boldsymbol{x})$. Using Theorem 2.1 with $\boldsymbol{c}_t - \lambda_t \boldsymbol{s}_t$ instead of $\boldsymbol{c}_t$, and $\widetilde{\boldsymbol{c}}_t - \lambda_t \boldsymbol{s}_t$ instead of $\widetilde{\boldsymbol{c}}_t$, we can write:

$$\sum_{t=1}^{T} \Big(f_t(\boldsymbol{x}^\star) - f_t(\boldsymbol{x}_t) + \lambda_t g_t(\boldsymbol{x}_t) - \lambda_t g_t(\boldsymbol{x}^\star) \Big) \leq \sqrt{2} D_{\mathcal{X}} \sqrt{h_{1:T}},$$

and rearrange to obtain:

$$R_T^{(e)} \leq \sqrt{2} D_{\mathcal{X}} \sqrt{h_{1:T}} + \sum_{t=1}^{T} \lambda_t g_t(\boldsymbol{x}^\star) - \sum_{t=1}^{T} \lambda_t g_t(\boldsymbol{x}_t). \tag{2.13}$$

For the dual update (2.11), we can use the non-proximal-FTRL bound [112, Theorem 1] to write:

$$-\sum_{t=1}^{T} \lambda_t g_t(\boldsymbol{x}_t) + \lambda \sum_{t=1}^{T} g_t(\boldsymbol{x}_t) \leq \frac{\lambda^2}{a_T} + \frac{1}{2} \sum_{t=1}^{T} a_t g_t^2(\boldsymbol{x}_t). \tag{2.14}$$

Since $g_t(\boldsymbol{x}^\star) \leq 0, \forall t$ and combining (2.13), (2.14) we get:

$$R_T^{(e)} \leq \sqrt{2} D_{\mathcal{X}} \sqrt{h_{1:T}} - \lambda \sum_{t=1}^{T} g_t(\boldsymbol{x}_t) + \frac{\lambda^2}{a_T} + \frac{1}{2} \sum_{t=1}^{T} a_t g_t^2(\boldsymbol{x}_t) \tag{2.15}$$

Setting $\lambda = 0$, using the identity:

$$\sum_{t=1}^{T} a t^{-\beta} \leq \frac{a T^{1-\beta}}{1-\beta}$$

and the bound $g_t(\boldsymbol{x}_t) \leq sJC$, we arrive at the $R_T^{(e)}$ bound.

For the violations, we use the following property in (2.15):

$$\frac{a_T}{2}\left[\sum_{t=1}^{T} g_t(\boldsymbol{x}_t)\right]_+^2 = \sup_{\lambda \geq 0}\left[\sum_{t=1}^{T} g_t(\boldsymbol{x}_t)\lambda - \frac{\lambda^2}{2a_T}\right],$$

Rearranging, we get:

$$\frac{a_T}{2}(V_T^{(e)})^2 \leq \sqrt{2}D_{\mathcal{X}}\sqrt{h_{1:T}} + \frac{a(sJC)^2}{2-2\beta}T^{1-\beta} - R_T^{(e)}.$$

Finally, taking the square root yields the $V_T^{(e)}$ bound. □

Discussion. The worst-case bounds in Theorem 2.4 arise when the predictions are failing. In that case, we have $\|\boldsymbol{c}_t - \widetilde{\boldsymbol{c}}_t\|^2 \leq 2w^2$ and use the bound $-R_T^{(e)} = \mathcal{O}(T)$ for the last term of $V_T^{(e)}$, to obtain $R_T^{(e)} = \mathcal{O}(T^{\kappa})$, with $\kappa = \max\{1/2, 1-\beta\}$ while $V_T^{(e)} = \mathcal{O}(T^{\phi})$, with $\phi = \frac{1+\beta}{2}$. Hence, for $\beta = 1/2$ we achieve the desired sublinear rates $R_T^{(e)} = \mathcal{O}(\sqrt{T}), V_T^{(e)} = \mathcal{O}(T^{3/4})$. However, when the RecSys manages to predict accurately the user preferences, the performance of π_{oec} improves substantially as the first terms in each bound are eliminated. Thus, for bounded T, we practically halve the regret and violation bounds.

It is also interesting to observe the tension between $V_T^{(e)}$ and $R_T^{(e)}$, which is evident from the $V_T^{(e)}$ bound and the condition $-R_T^{(e)} = \mathcal{O}(T)$. The latter refers to the upper bound of the *negative* regret, thus when it is consistently satisfied (i.e., for all T), we obtain an even better result: π_{oec} *outperforms* the benchmark. Another likely case is when $-R_T^{(e)} = \mathcal{O}(\sqrt{T})$, i.e., the policy does not outperform the benchmark at a rate larger than $\sqrt{T}$. Then, Theorem 2.4 yields $R_T^{(e)} = \mathcal{O}(T^{\kappa})$ with $\kappa = \max\{1/2, 1-\beta\}$ while $V_T^{(e)} = \mathcal{O}(T^{\phi})$ with $\phi = \max\{1/2, 1/4 + \beta/2\}$. Hence, for $\beta = 1/2$ the rates are reduced to $R_T^{(e)} = \mathcal{O}(\sqrt{T}), V_T^{(e)} = \mathcal{O}(\sqrt{T})$.

Finally, it is worth observing that π_{oec} can be readily extended to handle additional budget constraints such as time-average routing costs or average delays. And one can also generalize the approach to consider a budget-replenishment process where in each slot t the budget increases by an amount of b_t units. This is made possible due to the generality of the conditions (model perturbations can be non-stationary and correlated) under which the regret and violation bounds hold.

2.5. Caching with Multiple Predictors

In this section, we consider the case where we have additional predictors, apart from the RecSys, predicting the next-slot utility. Thus, we have a set of predictions $\{\widetilde{\boldsymbol{c}}_t^{(p)}, p \in \mathcal{P}\}$ at each slot t. To handle this setup and benefit from this abundance of predictions, we take a different approach and instead of using prediction-adaptive regularizers, as in the previous sections, we model the predictions as experts using the classical paradigm of learning through experts cf. [22]. Based on this approach,

we design a novel tailored optimistic *meta-learning* policy to accrue the best possible caching gains.

In particular, we associate an expert to each predictor, and we will abuse notation denoting them both with $p \in \mathcal{P}$. We refer to these predictors-linked experts as the *optimistic experts*. Every optimistic expert p proposes its caching action $\{\boldsymbol{y}_t^{(p)}\}_t$ at each slot t, by solving the following problem[12] :

$$\boldsymbol{y}_t^{(p)} = \arg\max_{\boldsymbol{y}\in\mathcal{Y}} \quad \widetilde{\boldsymbol{c}}_t^{(p)\top}\, \boldsymbol{y}. \tag{2.16}$$

Note that (2.16) is indeed a certainty-equivalent[13] linear program. We denote with $R_T^{(p)}$ the regret of each expert w.r.t the optimal-in-hindsight caching configuration for the entire time period of T slots, i.e.:

$$\boldsymbol{y}^\star = \arg\max_{y\in\mathcal{Y}} \boldsymbol{c}_{1:T}^\top\, \boldsymbol{y}.$$

Besides the optimistic experts, we consider an expert that *does not use predictions*. This special expert proposes an FTRL-based caching policy, and we refer to it as the *pessimistic expert* and associate it with the special index $p = 0$. The pessimistic expert proposes caching actions $\{\boldsymbol{y}_t^{(0)}\}_t$ according to eq. (2.6), but setting $\widetilde{\boldsymbol{c}}_t = 0$ for the regularization parameter σ_t in (2.5). Its regret is denoted with $R_T^{(0)}$. Our full experts set is the union of the group of optimistic experts with the pessimistic expert $\mathcal{P}^+ = \{0 \cup \mathcal{P}\}$.

We aim to learn a caching policy whose regret is upper-bounded by the regret of the best expert and does not exceed the $\mathcal{O}(\sqrt{T})$ regret of the pessimistic expert. Such a methodology of modeling policies as experts has been studied in the past [119]. However, this is the first work that implements optimistic learning through an experts model. In addition, here we also make the next step and propose to include a prediction for *the performance of each predictor*.

In particular, the caching decision is the convex combination of experts' proposals according to the *weights* $\boldsymbol{u}_t = (u_t^{(p)}, p \in \mathcal{P}^+)$ selected from the simplex:

$$\Delta_{\mathcal{P}^+} = \left\{ \boldsymbol{u} \in [0,1]^{P+1} \,\middle|\, \sum_{p\in\mathcal{P}^+} u_t^{(p)} = 1 \right\},$$

namely:

$$\boldsymbol{y}_t = \sum_{p\in\mathcal{P}^+} u_t^{(p)}\, \boldsymbol{y}_t^{(p)}. \tag{2.17}$$

Thus, $\boldsymbol{y}_{t+1}$ remains a feasible caching vector, despite being produced by mixing all the experts' proposals. The mixing weights, $\{\boldsymbol{u}_t\}_t$, are updated through a new

[12] To streamline the presentation, our analysis focuses on one cache, hence using only y_t decisions. However, this method can be readily extended to caching networks as discussed later.

[13] A certainty-equivalent program is one that considers a predicted utility vector as true and optimizes the decisions accordingly.

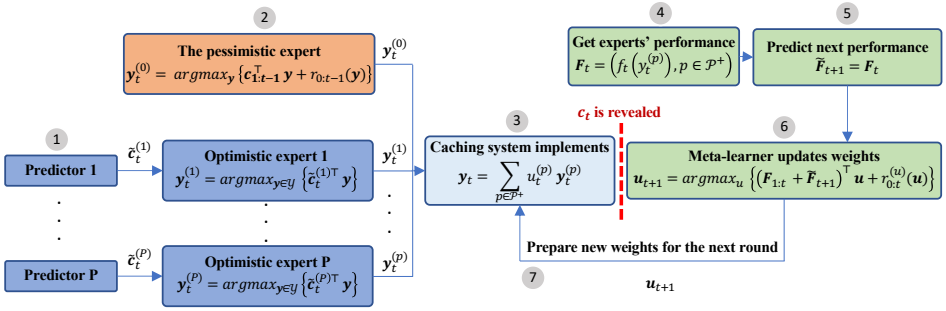


Figure 2.2: A decision step for the meta-learning policy π_{xc}. The policy is a combination of experts' proposals (predictors) according to their priority weights that were learned based on observations that are collected until slot t.

Algorithm 3: Experts Caching (π_{xc})

Input: C; $\boldsymbol{y}_1 \in \mathcal{Y}$; $\sigma = \sqrt{2}/D_{\mathcal{Y}}$.
Output: $\boldsymbol{y}_t$, $\forall t$.
for $t = 1, 2, \ldots$ **do**
  Observe utility predictions $\{\widetilde{\boldsymbol{c}}_t^{(p)}, p \in \mathcal{P}\}$
  Calculate optimistic proposals $\{\boldsymbol{y}_t^{(p)}\}_{p \in \mathcal{P}}$ with (2.16)
  Update $r_{0:t-1}(\boldsymbol{x})$ using (2.4)-(2.5) with $\widetilde{\boldsymbol{c}}_t = 0$
  Calculate pessimistic proposal $\boldsymbol{y}_t^{(0)}$ with (2.6)
  Serve request $\boldsymbol{q}_t$ with meta-policy $\boldsymbol{y}_t$ from (2.17)
  Observe the utilities of experts' proposals $\boldsymbol{F}_t = \left(f_t(\boldsymbol{y}_t^{(p)}), p \in \mathcal{P}^+\right)$
  Set experts performance prediction $\widetilde{\boldsymbol{F}}_{t+1} = \boldsymbol{F}_t$
  Calculate the new weights $\boldsymbol{u}_{t+1}$ using (2.18)
end

OFTRL step, which is similar to the updates of π_{obc} but we use the superscript (u) for the involved parameters to make clear the distinction. In particular, the update is:

$$\boldsymbol{u}_{t+1} = \arg\min_{\boldsymbol{u} \in \Delta_{\mathcal{P}^+}} \left\{ r_{0:t}^{(u)}(\boldsymbol{u}) - (\boldsymbol{F}_{1:t} + \widetilde{\boldsymbol{F}}_{t+1})^\top \boldsymbol{u} \right\}, \tag{2.18}$$

where $\boldsymbol{F}_t = \left(f_t(y_t^{(p)}), p \in \mathcal{P}^+\right)$ is the t-slot performance vector for the experts. The regularizers and the respective parameters, in this case, are decided by the following formulas:

$$r_0^{(u)}(\boldsymbol{u}) = I_{\Delta_{\mathcal{P}^+}}(\boldsymbol{u}), \quad r_t^{(u)}(\boldsymbol{u}) = \frac{\sigma_t^{(u)}}{2} \|\boldsymbol{u} - \boldsymbol{u}_t\|^2, \ t \geq 1,$$

$$\sigma_1^{(u)} = \sigma^{(u)}\sqrt{h_1^{(u)}}, \sigma_t^{(u)} = \sigma^{(u)}\left(\sqrt{h_{1:t}^{(u)}} - \sqrt{h_{1:t-1}^{(u)}}\right),\ t \geq 2,$$
$$\text{with} \quad h_t^{(u)} = \|\boldsymbol{F}_t - \widetilde{\boldsymbol{F}}_t\|^2.$$

For the predictions of the *experts' performance* $\tilde{F}_t$, at each time step, we will use experts' performance of the previous time step[14]:

$$\widetilde{\boldsymbol{F}}_t = \left(\widetilde{f}_t^{(p)},\ p \in \mathcal{P}^+\right) \triangleq \left(f_{t-1}(y_{t-1}^{(p)}),\ p \in \mathcal{P}^+\right).$$

This type of meta-optimism comes for free since, in practice, we expect the predictors to have consistent accuracy across contiguous slots; either accurately due to a recently trained model, or poorly due to e.g., distributional shift (see [120] and references therein) hence we can use the previous function. As for the pessimistic expert, its caching decisions do not vary much in consecutive slots (due to using strongly convex regularizers in updating the decisions).

The execution of the policy is summarized in Algorithm 3; and we also include the step-by-step visualization in Fig. 2.2. We see the sequence of steps where: (1) The Predictors output $\{\widetilde{\boldsymbol{c}}_t^{(p)}, p \in \mathcal{P}\}$; (2) The optimistic experts optimize for the predictions, whereas the pessimistic expert performs an FTRL step; (3) The experts' proposals are combined via the meta-learner weights; (4) The performance of experts' proposals is calculated through the revealed request; (5) The meta-learner optimistically sets the next-slot experts' performance to be the same as the current one; (6) & (7) The meta-learner performs an OFTRL step to calculate and set the weights for the next-slot. And the process repeats for the next slot.

The following theorem bounds the regret of the proposed meta-learning policy, which is defined as:

$$R_T^{(xc)} = \sum_{t=1}^{T} \boldsymbol{c}_t{}^\top(\boldsymbol{y}^\star - \boldsymbol{y}_t).$$

Recall that vector $\boldsymbol{c}_t$ indicates which file is requested and the utility associated with the request, and vectors $\boldsymbol{y}^\star$ and $\boldsymbol{y}_t$ are the optimal caching decisions/ the caching at time slot t, respectively.

Theorem 2.5. *Algorithm 3 ensures the regret bound:*

$$R_T^{(xc)} \leq 2\sqrt{\sum_{t=1}^{T}\|\boldsymbol{F}_t - \boldsymbol{F}_{t-1}\|^2} + \min_{p\in\mathcal{P}^+}\left\{R_T^{(p)}\right\} \leq 2w\sqrt{(P+1)T} + \min_{p\in\mathcal{P}^+}\left\{R_T^{(p)}\right\}.$$

[14] We note that the motivation for a large corpus of optimistic learning works stems from the fact that in many cases the cost functions are changing slowly [26, 93]. While the motivation in this work has been different until this section (availability of RecSys), the optimism in the meta-learner has the same scope as those initial works.

Proof. We first relate the regret of the combined caching decisions to that of the experts. Then, we can re-use the result of Theorem 2.1:

$$\begin{aligned} R_T^{(xc)} &= \sum_{t=1}^{T} \left(\boldsymbol{c}_t{}^\top \boldsymbol{y}^\star - \boldsymbol{c}_t{}^\top \sum_{p\in\mathcal{P}^+} u_t^{(p)}\ \boldsymbol{y}_t^{(p)} \right) = \sum_{t=1}^{T} \boldsymbol{c}_t{}^\top \boldsymbol{y}^\star - \boldsymbol{F}_t^\top \boldsymbol{u}_t \\ &= \sum_{t=1}^{T} \boldsymbol{c}_t{}^\top \boldsymbol{y}^\star - \boldsymbol{F}_t{}^\top \boldsymbol{u}^\star + \boldsymbol{F}_t{}^\top \boldsymbol{u}^\star - \boldsymbol{F}_t{}^\top \boldsymbol{u}_t \\ &= R_T^{(u)} + \min_{p\in\mathcal{P}^+} \left\{ R_T^{(p)} \right\} \end{aligned} \tag{2.19}$$

where $R_T^{(u)}$ is the regret for the *weights* $\boldsymbol{u}$: $R_T^{(u)} = \sum_{t=1}^{T} \boldsymbol{F}_t{}^\top \boldsymbol{u}^\star - \boldsymbol{F}_t{}^\top \boldsymbol{u}_t$. Note that (2.19) holds because $\boldsymbol{u}^\star = \arg\max_{\boldsymbol{u}}\ \boldsymbol{F}_{1:t}{}^\top \boldsymbol{u} = \boldsymbol{e}^{\boldsymbol{k}}$, $k = \arg\max_p f_t(\boldsymbol{y}_t^{(p)})$ and $\boldsymbol{e}^k$ is standard basis vector. Thus, we have:

$$\boldsymbol{F}_{1:t}^\top\ \boldsymbol{u}^\star = \max_{p\in\mathcal{P}^+} \left\{ \sum_{t=1}^{T} f_t(\boldsymbol{y}_t^{(p)}) \right\}.$$

We use the result of Theorem 1 to bound $R_T^{(u)}$:

$$R_T^{(u)} \le \sqrt{2} D_{\Delta_{\mathcal{P}^+}} \sqrt{\sum_{t=1}^{T} \|\boldsymbol{F}_t - \widetilde{\boldsymbol{F}}_t\|^2} \le 2w\sqrt{(P+1)T},$$

where the last inequality follows from the simplex diameter ($D_{\Delta_{\mathcal{P}^+}} \le \sqrt{2}$) and the fact that:

$$\|\boldsymbol{F}_t - \widetilde{\boldsymbol{F}}_t\|^2 \le \sum_{p\in\mathcal{P}^+} \|(f_t(\boldsymbol{y}_t^{(p)}) - f_{t-1}(\boldsymbol{y}_{t-1}^{(p)}))\|^2 \le |\mathcal{P}^+| w^2$$

i.e., we predicted a miss or a hit, whichever happened at $t-1$, but the opposite happens at t. Substituting in (2.19) gives the bound. □

Discussion. A key observation in Theorem 2.5 is that its bound contains the regret of the best optimistic expert. This yields very improved bounds for $R_T^{(xc)}$ whenever there is an optimistic expert that achieves negative regret. For example, if an expert can achieve[15] $R_T = \Theta(-T)$, e.g., because it has a very accurate RecSys (its recommendations are most-often followed), then the overall regret is $R_T^{(xc)} = \mathcal{O}(\sqrt{T}) - \Theta(-T)$, which becomes strictly negative for large T. Moreover, the $\mathcal{O}(\sqrt{T})$ term shrinks with more consistent performance of the experts, a condition that is rather expected in practical systems, thus getting us even faster to the regret of the best expert. Nonetheless, since the pessimistic expert exists in the group of experts, the min term is upper bounded by $\mathcal{O}(\sqrt{T})$ and $R_T^{(xc)}$ will maintain $\mathcal{O}(\sqrt{T})$ regardless of the performance of the optimistic experts.

[15] By this Θ notation we mean that $R_T = -cT$ for some absolute constant $c > 0$.

Since Algorithm 3 can work with a single optimistic expert, which is the setup handled by π_{obc}, it is interesting to compare their bounds. In fact, neither of those algorithms is better in all possible scenarios (regarding request patterns; evolution of utility parameters, etc. see also Sec. 2.6) than the other; and their relative performance ranking (in terms of utility) depends on the specific problem instance. For example, under worst-case predictions, we have that:[16]

$$\begin{aligned} R_T^{(xc)} &\leq 2w\sqrt{(P+1)T} + 2w\sqrt{CT} \\ &\leq 2(\sqrt{2}+\sqrt{C})w\sqrt{T} \qquad (P=1), \end{aligned} \tag{2.22}$$

which can be actually better than policy π_{obc}'s worst-case prediction bound[17] for practical values of the constants C.

On the other hand, in cases where inaccurate predictions occur for a certain fraction of the steps $\lceil \alpha T \rceil, 0 < \alpha < 1$ the min term in the $R_T^{(xc)}$ bound might evaluate to the pessimistic expert's regret since the optimistic experts can suffer linear regrets[18] $R_T^{(p)} \leq \mathcal{O}(\alpha T)$. For π_{obc}, the regret will be of the form

$$\begin{aligned} R_T &\leq 2\sqrt{C}\sqrt{\sum_{t\in[\lceil \alpha T \rceil]} \|\widetilde{\boldsymbol{c}}_t - \boldsymbol{c}_t\|} \\ &\leq 2\sqrt{2C}w\sqrt{\lceil \alpha T \rceil}. \end{aligned} \tag{2.23}$$

For example, for $\alpha \leq 1/2$, The upper bound in (2.23) is tighter than that in (2.22) for all C, w. Hence, π_{obc}'s regret can be smaller if in the described case. Overall, the choice between π_{xc} and π_{obc} in the case of a single predictor depends on the request sequence and the number of steps where predictions fail. Section 2.6 demonstrates these cases using various scenarios.

Regarding the extension to caching networks (more than one cache), note that sets $\mathcal{Y}$ and $\mathcal{Z}$ are convex by definition. Also, the set defined by the connectivity constraints (2.1) is convex (linear constraint in the variable x). Recalling that the Cartesian product and the intersection of convex sets is convex, we conclude that the constraint set $\mathcal{X}$ defined in (2.2), from which the joint caching-routing variable $\boldsymbol{z}$ is selected, remains convex. Finally, as demonstrated earlier, the meta-learner takes the convex combination of the experts' proposals, and since each expert proposes a caching-routing configuration $\boldsymbol{z}_t^{(p)} \in \mathcal{X}$, the meta caching-routing policy:

$$\boldsymbol{z}_{t+1} = \sum_{p\in\mathcal{P}^+} u_{t+1}^{(p)}\, \boldsymbol{z}_{t+1}^{(p)}, \qquad \boldsymbol{u}_{t+1} \in \Delta_{\mathcal{P}^+}$$

remains a valid one (i.e., $\boldsymbol{z}_t \in \mathcal{X}, \forall t$). Therefore, indeed, the ideas proposed in this section can be readily applied to bipartite caching networks.

[16] The pessimistic expert's regret is bounded by $2w\sqrt{CT}$ for the single cache setup (i.e., the diameter $D_{\mathcal{Y}} = \sqrt{2C}$).

[17] Note that π_{obc}'s worst-case bound is $2w\sqrt{2CT}$ for the single cache setup discussed here.

[18] This happens, e.g., when the pessimistic expert achieves a hit on the steps $[\alpha T]$.

Table 2.2: Online caching policies with *adversarial* guarantees: a summary of the contributions and comparison with the literature.

Alg.	**Model and Conditions**	**Guarantees** ($R_T, V_T \leq$)		**Adap. Learn.**
		Best case	**Worst case**	
1 (π_{obc})	• Bipartite network • Coded files • Predictions	0	$\mathcal{O}\left(\sqrt{T}\right)$	✓
2 (π_{ec})	• Bipartite • Coded files • Predictions • Budget constr.	$\mathcal{O}\left(\frac{\kappa}{2}\sqrt{T}\right), \mathcal{O}\left(\frac{\kappa}{2}T^{\frac{3}{4}}\right)$	$\mathcal{O}\left(\kappa\sqrt{T}\right), \mathcal{O}\left(\kappa T^{\frac{3}{4}}\right)$	✓
3 (π_{xc})	• Bipartite network • Coded files • $P \geq 1$ predictors	$A_P \triangleq \min_{p\in\mathcal{P}^+}\left\{R_T^{(p)}\right\}$	$2w\sqrt{(P+1)T}$ + A_P	✓
[57]	• Single cache • Coded & uncoded files	$\mathcal{O}\left(\sqrt{T}\right)$		–
[55]	• Bipartite network • Coded files	$\mathcal{O}\left(\sqrt{T}\right)$		–
[58]	• Bipartite network • Coded (single cache) & uncoded	$\mathcal{O}\left(\sqrt{T}\right)$		–
[56]	• General graph network • Coded & uncoded files	$\mathcal{O}\left(\sqrt{T}\right)$		–
[59]	• Bipartite network • Coded & uncoded files	$\mathcal{O}\left(\sqrt{T}\right)$		–

Technical comparison with the literature. Now that we have presented all of our algorithms, let us summarize in Table 2.2 their main features and revisit how they compare with the state-of-the-art results. For Alg. 1 - 2, the best case refers to the scenario where the request predictions are perfect $\widetilde{\boldsymbol{c}}_t = \boldsymbol{c}_t, \forall t$; and the worst case to the scenario where predictions are furthest from the truth $\widetilde{\boldsymbol{c}}_t = \arg\max_{\boldsymbol{c}} \|\boldsymbol{c} - \boldsymbol{c}_t\|, \forall t$. The dependence of the constant factors of the regret bound, denoted with κ to facilitate presentation, was made explicit for Alg. 2 where we saw that these constants shrink with the predictions' accuracy; and the same holds for Alg. 1. For Alg. 3, the best case refers to the scenario where the experts' predictions are perfect $\widetilde{\boldsymbol{F}}_t = \boldsymbol{F}_t, \forall t$; while the worst case arises when $\widetilde{\boldsymbol{F}}_t = \arg\max_{\boldsymbol{F}} \|\boldsymbol{F} - \boldsymbol{F}_t\|, \forall t$. Algorithms that do not leverage predictions in regret analysis (all prior work in caching[19]) have the best and worst case columns merged. We also distinguish between adaptive and static learning rates. While some prior works do employ

[19] We note the exception of [70] which considers *ML advice* in the paging problem (single cache, uncoded), but its bounds are defined w.r.t. the cache size and quantified in terms of competitive ratio – a different metric than regret, see discussion in [74].

time-adaptive learning (dynamic steps), as e.g., in [55], none of them adapts the rates (or, equivalently the regularization) to the observed gradients $\{\boldsymbol{c}_t\}_t$ and/or prediction errors, as we propose here, but instead use the Lipschitz constant w, where $\|\boldsymbol{c}_t\| \leq w, \forall t$. This leads to looser bounds in most practical cases [112] and, of course, does not allow to benefit from the availability of predictions.

2.6. Performance Evaluation

We evaluate π_{obc}, π_{oec} and π_{xc} under different request patterns and predictions modes; and we benchmark them against $x^\star$ and the OGD policy [55] that outperforms other state-of-the-art policies [108, 109]. Note that the OGD policy has been shown to match the theoretical lower bound on the regret [55, Thm. 1]. In other words, it achieves (up to constant factors) as small regret as theoretically possible under adversarial settings and *without* predictions. Hence, it serves as the main competitor in our experiments. We observe that when reasonable predictions are available, the proposed policies have an advantage, and under noisy predictions, they still reduce the regret at the same rate with OGD, as proven in the Theorems. First, we compare π_{obc} and π_{xc} against OGD [55] in the single cache case. We then study π_{obc} for the bipartite model and π_{oec} with the presence of budget constraints. We consider three requests scenarios, stationary Zipf requests (with parameter $\zeta = 1.1$) and two actual request traces: YouTube (YT) [104] and MovieLens (ML) [105]. For predictions, we assume that at each time step, the user follows the recommendation with probability ρ (unknown to the caching system), and we experiment with different ρ values. The full codebase for the proposed policies and experiments is available via GitHub [121].

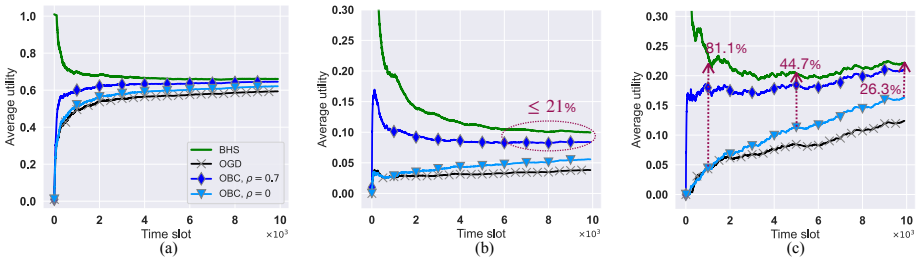


Figure 2.3: Utility in the single cache model with one RecSys of different recommendation quality levels (i.e., ρ) in (a) Zipf requests with $\zeta = 1.1$, (b) YouTube request traces, (c) MovieLens request traces.

Single Cache Scenarios. We set $w=1$ to study the cache hit rate scenario, use a library size of $N = 10^4$ files, and cache capacity of $C = 100$ files. Figures 2.3.a-c. depict the attained average utility, $\frac{1}{t}\sum_{i=1}^{t} f_i(\boldsymbol{x}_i)$, for each policy and the Best in Hindsight (BHS) cache configuration *until that slot*, i.e., we find the best in hindsight[20] for each t. Note that BHS always achieves utility 1 initially (first requests to fill the cache). Thus, we cut the y-axis for better presentation in Figures 2.3.b,c.

[20] Unlike [55] that finds $x^\star$ for $t = T$, we find a $x_t^\star$ for each t. Thus, the gap among any policy and BHS is the *evolving* average regret R_t/t.

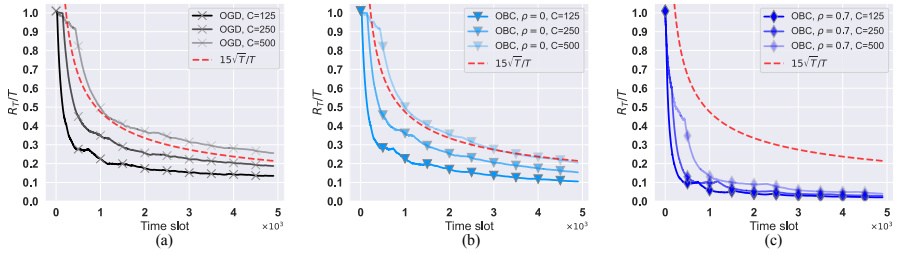


Figure 2.4: Regret over time in the single cache model with different values of the cache capacity for (a) π_{ogd}, (b) π_{obc} with $\rho = 0$, (c) π_{obc} with $\rho = 0.7$.

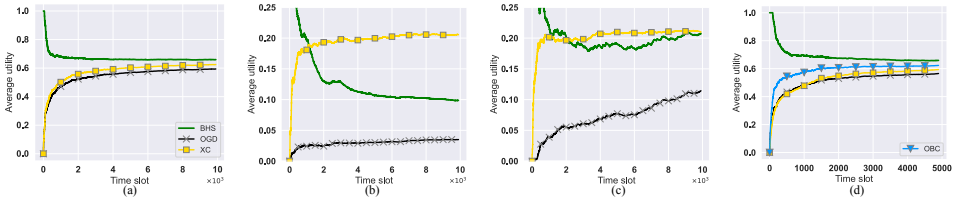


Figure 2.5: Utility in the single cache model with two RecSys of recommendation qualities $\rho = 2\%$ and $\rho = 20\%$, each modeled as an expert within XC, in (a) Zipf requests with $\zeta = 1.1$, (b) YouTube request traces, (c) MovieLens request traces. (d) A comparison between π_{obc} and π_{xc} using one RecSys of an alternating recommendation quality.

It can be seen that the accurate predictions (i.e., when users follow the recommendations 70% of the time) push the performance of our online caching towards BHS. For example, in Fig. 2.3.b, the utility gap is at most 21% after $t = 6k$. At the same time, even when users do not follow the recommendation at all, we still maintain a diminishing regret; the gap in Fig. 2.3.c goes from 81.1% at $t = 1k$, to 26.3% at $t = 10k$. In Fig. 2.4, we study the effect of increasing the cache size. Expectedly, the regret increases since the diameter of the decision set also increases. However, with higher prediction quality, this effect is minimized since the regret is proportional to a shrinking error term. We also plot $15\sqrt{T}/T$ which is proportional to the average theoretical regret bound (Recall that $R_T \propto \sqrt{CT}$), and remark that the average regret of the proposed algorithms decays at a similar rate.

For multiple predictors, we use π_{xc}. In Figures 2.5.a-c, we evaluate π_{xc} with 3 experts: an FTRL expert, and two other optimistic experts. The first optimistic expert is endowed with a predictor (e.g., a recommendation system) that gets followed with probability $\rho = 2\%$. For the other it is $\rho = 20\%$. As shown in the plots, π_{xc} achieves negative regret on the traces (it outperforms the BHS policy) and converges to the performance of the best expert (0.20 utility). This is because in more spread distributions and real request traces, predicting the next request provides a great advantage for policies that modify the cache online over the fixed BHS. In the stationary Zipf request pattern, the optimal cache is for the files with top probabilities, which are easily captured by BHS. Thus, BHS policy performs the best. In Fig. 2.5.d we show the advantages of π_{obc} compared to π_{xc} *with two experts*:

an FTRL expert and a recommendation-based expert. ρ alternates between 100% and 0% (i.e., requesting the file recommended at a time step t, and any other file at $t+1$, and so on). Here, π_{obc} outperforms π_{xc} since the alternating prediction accuracy induces frequent switching between the two experts in π_{xc}: the performance of the optimistic expert alternate between 0 and 1, while that of pessimistic expert is in the range $(0.55, 0.65)$. Hence, π_{xc} is inclined to place some weight on the prediction expert at one step, only to retract and suffer a greater loss at the following one had it stayed with the full weight on the FTRL expert. Due to the additional regret caused by such frequent switching, π_{obc}'s regret is 54.8% of π_{xc}'s. π_{obc} also achieves 38.2% of π_{ogd}'s regret. It is noteworthy that the optimal *dynamic* policy would achieve a utility of 1 across all time steps. This is because, unlike the BHS policy, it is allowed to change at every time step and can thus simply cache every file before it gets requested (recall that the optimal dynamic caching policy is defined as $\boldsymbol{x}_t^\star = \arg\max_{\boldsymbol{x}\in\mathcal{X}} f_t(\boldsymbol{x})$). However, as mentioned earlier when introducing the benchmark, it is unclear whether such a policy is desirable since it comes with a high switching cost. Furthermore, under the assumptions in this paper, no algorithm can find such a policy beforehand.

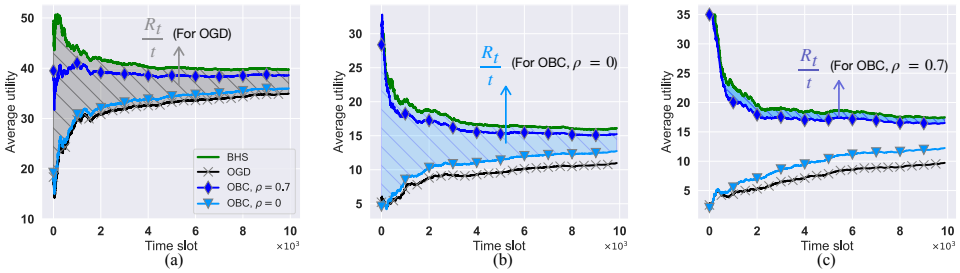


Figure 2.6: Attained utility in the bipartite model under different recommendation quality levels in (a) Zipf requests with $\zeta = 1.1$, (b) YouTube request traces, (c) MovieLens request trace.

Bipartite Networks. We consider next a bipartite graph with 3 caches and 4 user locations, where the first two locations are connected with caches 1 and 2, and the rest are connected to caches 2 and 3. The utility vector is $w_n = (1, 2, 100), \forall i, j$, thus an efficient policy places popular files on cache 3. This is the setup used in [55] that we adopt here to make a fair comparison. For the zipf scenario, we consider a library of $N = 1000$ files and $C = 100$. For the traces scenario, files with at least 10 requests are considered, forming a library of $N = 456$ files for the YouTube dataset, and we set $C = 50$, and $N = 1152$ for the ML dataset, and we increase $C = 100$. The location of each request is selected uniformly at random. Similar to the single-cache case, we plot the average utility of the online policies and the best static configuration *until each* t. Recall that the area between a policy and BHS is the average regret of that policy. To avoid clutter, we shade this area for OGD in the first sub-figure, as an example, and for OBC in the next two.

In Fig. 2.6.a, the effect of good predictions is evident as OBC maintains utility within 5.5% of BHS's utility after $t = 2.5k$. Even when the recommendations are not followed, OBC preserves the sublinear regret, achieving a gap of 30.4% and

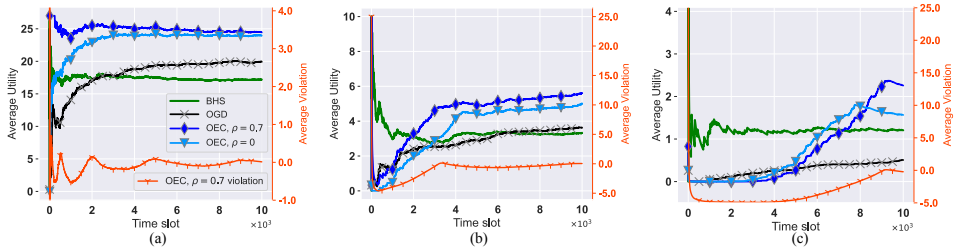


Figure 2.7: Utility and budget utilization with (a): Zipf requests with $\zeta = 1.1$ and (b): YouTube traces (c): MovieLens traces.

8.5% for $t = 1k$ and $t = 10k$, respectively. Akin patterns appear in the traces scenarios. Namely the similarity between OBC with good predictions and BHS, and the improvement in OBC utility despite the recommendations quality. We also note lower utility across all policies due to the more spread requests. Note that under this bipartite model, the optimal dynamic policy, $\boldsymbol{x}_t^\star$, would achieve a utility of $\exp f_t(\boldsymbol{x}_t^\star) = 4/4 + 200/4 = 51$. This is because the requests appear uniformly randomly at one of the four users' locations, and the first two locations can always be served by cache 2 (utility 2), whereas the other two locations can be served by cache 3 (utility 100).

Next, we consider the case of budget constraint and evaluate π_{oec} for Zipf requests in Fig. 2.7.a, and the traces in Fig. 2.7.b, c. The prices at each t are generated uniformly at random in the normalized range $[0, 1]$, and the available budget is generated randomly $b_t = \mathcal{N}(0.5, 0.05) \times 10$ i.e., enough for approximately 10 files. Such tight budgets magnify the role of dual variables and allow testing the constraint satisfaction. The benchmark $\boldsymbol{x}^\star$ is computed once for the *full time horizon*, and its utility is plotted for each t. In both scenarios, we note the constraint violation for all policies is similar, fluctuating during the first few slots and then stabilizing at zero. Hence, we plot it for one case.

Concluding, we find that π_{oec} can even outperform the benchmark. This is because the actual request patterns in the traces are not actually adversarial. Also, unlike the benchmark policy, π_{oec} is allowed to violate the budget at some time slots, provided that the constraints are eventually satisfied, which occurs either due to strict satisfaction or due to having an ample subsidy at some slots. For example, in the first scenario (Fig. 2.7.a), the good predictions enable OEC to outperform $x^\star$ by 42.2% after observing all requests ($T = 5K$). OGD, and OEC with noisy predictions attain utility units improvement of 16.1%, 39.3%, respectively, over the BHS. We note in Fig.2.7.b, c the delay in learning due to initially over-satisfied budget constrain. This is a transient effect as the dual variables approach their optimal value. Eventually, the bounds on regret are satisfied. We stress that the algorithms scale for very large libraries $\mathcal{N}$; the only bottleneck in the simulations is finding $x^\star$, which involves the horizon T; this is not required in real systems.

Computational Complexity. Note that the OFTRL update (e.g., (2.6)) is either a linear program in case $r_t(\cdot) = 0, \forall t$, or a quadratic program otherwise. In the former case, the solution is finding the top C most requested files. This can

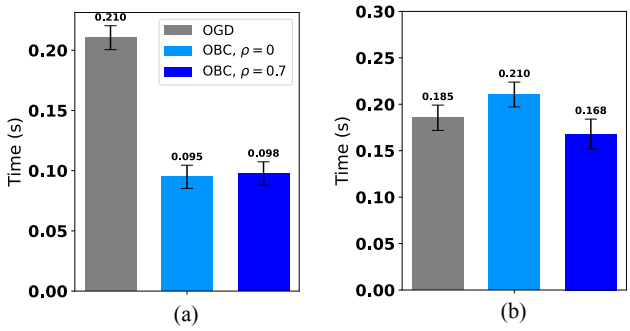


Figure 2.8: Average time consumed per decision step in the bipartite network configuration and the ML Dataset ($N = 1152$, $C = 100$, $|\mathcal{I}| = 4$, $|\mathcal{J}| = 3$) with (a) pre-reserved storage (π_{obc}), and (b) elastic storage (π_{oec}).

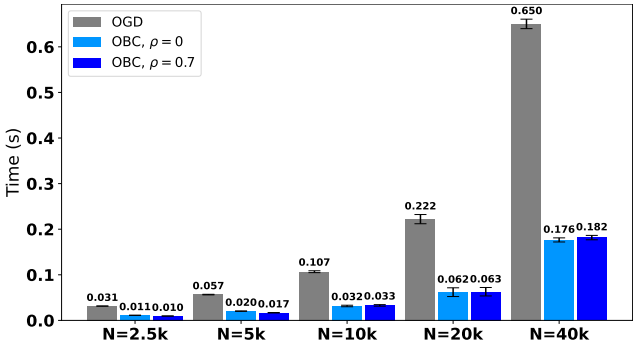


Figure 2.9: Average time consumed per decision step in the single cache configuration and stationary requests with different library sizes N. The sample size is $T = 10k$

be done through a simple ordering operation whose worst-case complexity is linear in the dimension of the decision variable (e.g., $\mathcal{O}(N \log(N)$ for each cache). If, however, we have regularization terms, then the resulting quadratic program can be solved in closed form in $\mathbb{R}$. Then, for the projection to the feasible set $\mathcal{X}$, we can use specialized projection algorithms for the capped simplex (the capacity constraints for each cache) whose worst-case complexity is still polynomial in the dimension (i.e., $\mathcal{O}(N^2)$)[122].

The above discussion refers to worst-case scenarios. In practice, we leverage the fact that we repeatedly solve similar optimization problems at each time step (the update step differs by adding one linear term and one quadratic term). Thus, the solution in a step can be used as an initial point for the following one. We note that the experimental running times are significantly less than the worst-case ones, as can be seen in Fig. 2.8 and Fig. 2.9. These simulations were performed using `CVXPY` 1.2 package with `Python` 3.10 running on an Apple M1 Pro Chip and 16GB of RAM. The difference between OGD and the optimistic one is due to the difference in the update problem structure (lazy vs. greedy projection). The chosen library

sizes were selected considering that the simulations are done on a conventional PC, yet they do provide insight into the scalability, which is mostly positive as discussed earlier. Larger library sizes might necessitate custom hardware setup, or heuristic techniques such as dividing the library into sub-sets and running one of the proposed algorithms on each subset.

Batched Requests. As mentioned in the discussion of the system model, as well as that of Theorem 1, our assumption on the request model (one request per time slot) is for technical ease of analysis. The main feature of the proposed policies, which is having $R_T \propto \mathcal{O}(\sqrt{\sum_{t=1}^{T} \|\boldsymbol{c}_t - \widetilde{\boldsymbol{c}}_t\|})$, remains valid when the request model is batched (i.e., processing B requests per time slot). However, the upper bound (not necessarily the regret) will be scaled accordingly since the diameter of the decision set will now increase. Namely, following the same steps in the proof of Theorem. 1 (after (2.8)), instead of $D_{\mathcal{X}} = 2(JC+1)$, we would have $D_{\mathcal{X}} = 2(JC+B)$. Fig. 2.10 shows experimentally how the batch size affects the regret. We also plot the line ${}^{18\sqrt{T}}/_{T}$ (recall that $R_T \propto G\sqrt{T}$, and in turn $G \propto B$). In this experiment, we introduce a new parameter, $\bar{\rho}$, which denotes the percentage of requests correctly predicted out of the total B. We note in these experiments that with better predictions, the effect is amortized since the term $\sqrt{\sum_{t=1}^{T} \|\boldsymbol{c}_t - \widetilde{\boldsymbol{c}}_t\|}$ remains small.

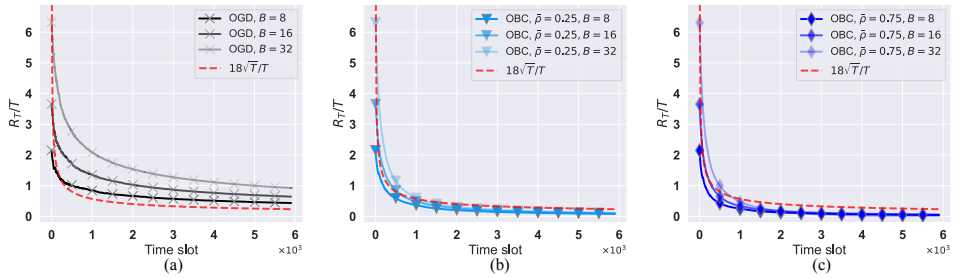


Figure 2.10: Average regret over time for the single cache, generalized batched requests model for (a) π_{ogd}, (b) π_{obc} with $\bar{\rho} = 0.25$, and (c) π_{obc} with $\bar{\rho} = 0.75$.

2.7. Conclusion

The problem of online caching is timely with applications that extend beyond content delivery to edge computing and in fact to any dynamic placement problem with Knapsack-type constraints. This chapter proposes a new suite of caching policies that leverage predictions obtained from content recommendations, and possibly other forecasters, to minimize the caching regret w.r.t an ideal (yet unknown) benchmark cache configuration. As recommender systems permeate online content viewing platforms, such policies can play an essential role in optimizing caching efficacy. We identified and built upon this new connection between caching and recommender systems. The proposed algorithmic framework is scalable and robust to the quality of recommendations and the possible variations of network state and the request sequences, which can even be decided by an adversary. The achieved

bounds improve upon the previously known caching regret performance, see [55, 56, 57, 58] and references therein. Finally, we believe this work opens new research directions both in terms of caching, e.g., pursuing the design of optimistic policies for uncoded caching; and in terms of resource scheduling in pertinent network and mobile computing problems using untrusted sources of optimism, i.e., predictors of unknown or varying accuracy.

One key limitation of the methods discussed in this chapter is the assumption that files can be divided into arbitrarily small chunks, i.e., that caching decisions are continuous. While this abstraction is appropriate for modeling coded caching systems, it does not hold in scenarios where files must be stored in their entirety, which is both common and practically important. The next chapter addresses this limitation by extending the optimistic learning framework to whole-file caching and, more broadly, to knapsack-style problems, where decisions are inherently discrete and combinatorial.

3

Optimistic Learning for Discrete Domains: Whole-file Caching

In the previous chapter, we addressed the caching problem under the assumption that files could be divided into arbitrarily small chunks. This corresponds to the coded caching setting, which, from a theoretical perspective, allows the caching decision space to be continuous, enabling efficient and optimal solutions. In this chapter, we relax this assumption and focus on the discrete caching setting, where each file must be stored in its entirety or not at all. As noted in the introduction, this shift has significant implications both practically and theoretically. Practically, the problem generalizes to selecting the best k out of m possible objects—a setting that arises ubiquitously in communication networks and computing systems. Theoretically the resulting problem is no longer convex, necessitating a fundamentally different set of algorithmic and analytical tools. The goal of this chapter is to design robust *discrete* caching policies that are able to learn effective caching decisions with the aid of a prediction oracle of *unknown quality* (Fig. 3.1 left) even when the file requests are made in an adversarial fashion.

To that end, we formulate again the caching problem as an online convex optimization (OCO) problem [22]. At each slot $t = 1, 2, \ldots, T$, a learner (the caching

The content of this chapter has been published in:

- Naram Mhaisen, Abhishek Sinha, George Paschos, and George Iosifidis. "Optimistic No-regret Algorithms for Discrete Caching". In *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 6.3 (2022), pp. 1-28.
- Naram Mhaisen, Abhishek Sinha, George Paschos, and George Iosifidis. "Optimistic No-regret Algorithms for Discrete Caching". In *Abstract Proceedings of ACM SIGMETRICS 2023*.

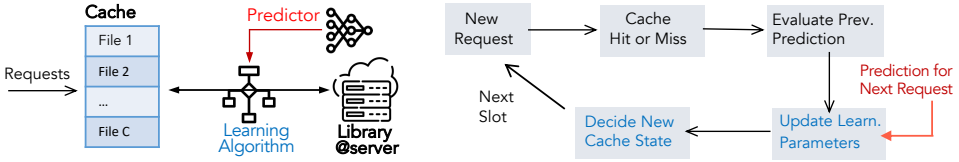


Figure 3.1: Optimistic online caching with predictions: system schematic (*left*) & algorithm template (*right*).

3

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 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. 3.1 right). As customary in the online learning literature, we characterize the policy's performance by using the static *regret* metric. Recall:

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

where $x^\star = \arg\max_{x \in \mathcal{X}} \sum_{t=1}^T f_t(x)$ 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 $\mathcal{O}(\sqrt{T})$ regret bound [58, 62, 63, 64, 65, 75, 123], which, in fact, is the optimal (as small as possible) achievable regret rate, see [28, Thm. 5.1], [58, Thm. 1].

Most of these regret-optimal algorithms are 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 [124]. 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 handy 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

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

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 [58, 64, 65]. In this chapter, we extend their scope and design algorithms with substantially improved performance guarantees.

Chapter notation. We denote sets with calligraphic capital letters, e.g., $\mathcal{N} = \{1, 2, \ldots, N\}$; vectors with small letters, e.g., $x = (x_i, i \in \mathcal{N})$ 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.

3.1. 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 out 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. 3.3). 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. 3.2.1), whose regret match this lower bound up to constants. In detail, we expand the optimistic FTRL algorithm [26, 75, 123] that was designed for convex problems, to handle, through sampling, discrete (non-convex) decisions (Sec. 3.4). We prove this approach attains expected regret $\mathcal{O}(\sqrt{T})$ 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 ($\mathcal{O}(N)$) computation overhead (Sec. 3.5). The flip side is that its regret bound contains an $\mathcal{O}(\textit{poly-log}(N))$ term.

We first derive results for equal-sized files, in line with all prior learning-based works for discrete caching [58, 65, 70, 71] and continuous caching [62, 64, 75]. Subsequently, we drop this assumption and study the single cache problem with different file sizes (Sec. 3.6). These first-of-their-kind regret-based algorithms require a 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 a rounding subroutine, `DepRound` [127], to a known almost-discrete optimal solution `Dantz` [128]. We show that the proposed policies

[2] Optimistic learning was originally proposed for problems with slowly-varying (hence, predictable) cost functions [125]; 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 $\mathcal{O}(N^2)$ even for the non-weighted capped simplex [126].

achieve (1/2)-approximate regret of $\mathcal{O}(\sqrt{T})$ and zero-regret for adversarial and perfect predictions, respectively. We also extend the OFTRL analysis to the widely used bipartite network caching model [30, 32] (Sec. 3.7), where we optimize jointly the discrete caching and routing decisions to obtain prediction-modulated performance.

In (Sec. 3.8), 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 suggests 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 (performance of the optimistic expert) in a way that achieves negative regret when that expert is reliable, and, again, maintains an $\mathcal{O}(\sqrt{T})$ 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.

3.2. Chapter Background

In this section, we introduce some technical background and related works on optimistic online learning for discrete domains.

3.2.1. Adaptive Smoothing

Our optimistic learning approach is based on *adaptive smoothing*. Abernethy *et. al.* [129] 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:

$$x_t = \arg\max_{x\in\mathcal{X}}\langle\Theta_{t-1}, x\rangle = \nabla\Phi(\Theta_{t-1}),$$

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 [130, 131], since successive gradients of non-smooth functions can be arbitrarily far from each other, thus leading to unstable actions. [129] 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] $x_t = \arg\max_{x\in\mathrm{conv}(\mathcal{X})}\langle x,\Theta_t\rangle - r_{1:t}(x)$

[4] While the maximization requires that x_t to be in the convex hull of $\mathcal{X}$, feasibility can be recovered via appropriate rounding.

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 [62] and [64] respectively[5]. As for FTPL, the smoothing is done by adding perturbation to the accumulated cost parameter of the potential. 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 [58]. 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 less regularization/perturbation (less smoothing), enabling the learner to align its decisions more with 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 $R_T \leq \mathcal{O}(\sqrt{T})$. Nonetheless, these two algorithms have considerable differences in terms of computational complexity and constants in the bounds, which are discussed in detail.

3.2.2. Discrete Optimistic Learning

OFTRL versions have been proposed in [26] and recently used in [132] for problems with budget constraints, while [75, 123] tailored these ideas to *continuous* caching. The problem of discrete caching is fundamentally different. Through careful analysis, we manage to reuse these results after relaxing the cache integrality constraints and then employing 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 [133] and [134]. In [133], 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 [134], 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 $\mathcal{O}(\log(N)^{1/4})$, 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 is available in these prior works. We kindly refer the reader to table 3.1 for an overview of the presented algorithms

[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 [112, 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 $\nabla\Phi(\Theta_t + \eta_t\gamma) = \mathbb{E}_\gamma[x_t]$

in the context of the most related literature.

3.3. 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 [26, 92, 125], 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 efficient training of the forecaster using a convex loss function (please see Appendix 3.9 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 3.1. *For any online caching policy π, there exist a sequence of requests $\{\theta_t\}_T$ and predictions $\{\tilde{\theta}_t\}_T$ for which the regret R_T satisfies*

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

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 [135] 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 predictions $\tilde{\xi}_t$ are also chosen independently and uniformly at random from the probability simplex Δ_N. Specifically, we let

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

[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 [62, 123].

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 that $\xi_t \perp \left(\{\tilde{\xi}_\tau\}_{\tau=1}^t, \{\xi_\tau\}_{\tau=1}^{t-1}, X_t^\pi\right)$ 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 $\mathbb{E}[\langle \xi_t, X_t^\pi\rangle] \leq \frac{1}{2}$. 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 $\mathbb{E}\big[\sum_{t=1}^T \langle \xi_t, X_t^\pi\rangle\big] \leq \frac{T}{2}$.

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 [58], 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 [58, Lemma 1]:

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

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

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

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.

$$\begin{aligned}
\mathbb{E}\left[M_T\right] &= \mathbb{E}\left[\sum_{t=1}^T \|\tilde{\xi}_t - \xi_t\|_2^2\right] \overset{(a)}{=} T\ \mathbb{E}\left[||\tilde{\xi}_1 - \xi_1||_2^2\right] \overset{(b)}{=} TN\ \mathbb{E}\left[(\tilde{\xi}_1^1 - \xi_1^1)^2\right] \\
&= TN\ \mathbb{E}\left[(\tilde{\xi}_1^1)^2 - 2\tilde{\xi}_1^1\xi_1^1 + (\xi_1^1)^2\right] \\
&\overset{(c)}{=} TN\left[\mathrm{Var}(\tilde{\xi}_{1,1}) + (\mathbb{E}(\tilde{\xi}_{1,1}))^2 - 2\mathbb{E}(\tilde{\xi}_{1,1})\mathbb{E}(\xi_{1,1}) + \mathbb{E}(\xi_{1,1}^2)\right] \\
&\overset{(d)}{=} TN\left[\frac{(N-1)}{N^2(N+1)} + \frac{1}{N^2} - \frac{2}{N^2} + \frac{1}{N}\right] = T\big(1 - \frac{2}{N(N+1)}\big) \leq T,
\end{aligned}$$

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 (3.2), we have by Jensen's inequality

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

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

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

We will see that the proposed optimistic algorithms in Sections 3.4 and 3.5 attain this bound within an absolute and a poly-logarithmic factor, respectively.

3.4. Caching through Optimistic Regularization (OFTRL-Cache)

The first algorithm we propose is based on OFTRL. Prediction adaptive regularization was explored before in [125] and later improved via proximal regularizers in [26], all for convex sets. 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 [136] to recover integral caching vectors $x_t \in \mathcal{X}, \forall t$, which satisfy the hard capacity non-convex constraint. In other words, we define:

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

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 [123].

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:

$$r_t(x) = \frac{\sigma_t}{2}\|x - x_t\|_2^2. \tag{3.3}$$

Following [75], 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 on the predictor's accuracy.

Algorithm 4: Optimistic Follow The Regularized Leader (`OFTRL-Cache`)

1 **Input**: $\sigma = 1/\sqrt{C}$, $\delta_1 = \|\theta_1 - \tilde{\theta}_1\|_2^2$, $\sigma_1 = \sigma\sqrt{\delta_1}$, $x_1 = \arg\min_{x\in\mathcal{X}}\langle x, \theta_1\rangle$
2 **Output**: $\{x_t \in \mathcal{X}\}_T$ *// Feasible discrete caching vector at each slot*
3 **for** $t = 2, 3 \dots$ **do**
4 $\quad\tilde{\theta}_t \leftarrow$ `prediction` *//Obtain request prediction for slot t*
5 $\quad\hat{x}_t = \arg\max_{x\in\text{conv}(\mathcal{X})} \left\{-r_{1:t-1}(x) + \langle x, \Theta_{t-1} + \tilde{\theta}_t\rangle\right\}$ *// Update the continuous cache vector*
6 $\quad x_t \leftarrow MadowSample(\hat{x}_t)$ *// Obtain the discrete cache vector using Algorithm 13*
7 $\quad\Theta_t = \Theta_{t-1} + \theta_t$ *// Receive t-slot request and update total gradient*
8 $\quad\sigma_t = \sigma\left(\sqrt{\delta_{1:t}} - \sqrt{\delta_{1:t-1}}\right)$ *// Update the regularization parameter*
end

The detailed steps are summarized in Algorithm 4. 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 3A.1). 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 4 is described next.

Theorem 3.2. *Algorithm 4 ensures, for any time horizon T and $N \geq 2C$, 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 [75, Lem. 1], which we restate below:

Lemma 3.3. *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 4 guarantee the bound $\hat{R}_T \leq r_{1:T}(x^\star) + \frac{1}{2}\sum_{t=1}^{T}\|\theta_t - \tilde{\theta}_t\|_{(t),\star}^2$.*

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 $\|x^\star - x_t\|_2 \leq \sqrt{2C}$,

[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.
[10] N is typically orders of magnitude 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 $\sqrt{2(N-C)}$.

$\forall x^\star, x_t \in \text{conv}(\mathcal{X})$. Thus, we can upper-bound the regularizers in (3.3), replace in the above Lemma and telescope to get:

$$\hat{R}_T \leq \sigma_{1:T}\, C + \frac{1}{2}\sum_{t=1}^{T}\frac{\delta_t}{\sigma_{1:t}}. \tag{3.4}$$

Observing that the sum $\sigma_{1:t}$ telescopes to $\sigma\sqrt{\delta_{1:t}}$, we can substitute it in (3.4) and use the standard identity [28, Lem. 4.13] to bound the second term via $\sum_{t=1}^{T}\delta_t/\sqrt{\delta_{1:t}} \leq 2\sqrt{\delta_{1:T}}$. Therefore, we obtain: $\hat{R}_T \leq \sigma\sqrt{\delta_{1:T}}\; C + \frac{1}{\sigma}\sqrt{\delta_{1:T}}$, and by setting the parameter σ to its optimal value $\sigma = 1/\sqrt{C}$, we can eventually write:

$$\hat{R}_T \leq 2\sqrt{C}\sqrt{\sum_{t=1}^{T}\|\theta_t - \tilde{\theta}_t\|_2^2}. \tag{3.5}$$

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 $\Pr(x_i^t = 1) = \pi_i - \pi_{i-1} = \hat{x}_i^t$, 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 $\mathbb{E}[x_t] = \Pr(x_i^t = 1) = \hat{x}_t$. The result follows by using (3.5) and observing:

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

□

Discussion. The bound in Theorem 3.2 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 $R_T \leq 0$; when all predictions fail, we get $R_T \leq 2\sqrt{2CT}$. 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 [112, Sec. 3.4]), and ~ 5 compared to the lower bound derived in Sec. 3.3. 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 [22, 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.

[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 is limited.

Algorithm 5: Optimistic Follow The Perturbed Leader (`OFTPL-Cache`)

Input: $\eta_1 = 0$, $y_1 = \arg\min_{y \in \mathcal{X}} \langle y, \theta_1 \rangle$
Output: $\{y_t \in \mathcal{X}\}_T$ *// Feasible discrete caching vector at each slot*
$\gamma \overset{iid}{\sim} \mathcal{N}(0, \mathbf{1}_{N \times 1})$ *// Sample a perturbation vector*
for $t = 2, 3, \ldots$ **do**
  $\tilde{\theta}_t \leftarrow$ `prediction` *//Obtain request prediction for slot t*
  $\eta_t = \frac{1.3}{\sqrt{C}} \left(\frac{1}{\ln(Ne/C)} \right)^{\frac{1}{4}} \sqrt{\sum_{\tau=1}^{t-1} \|\theta_\tau - \tilde{\theta}_\tau\|_1^2}$ *// Update the perturbation parameter*
  $y_t = \arg\max_{y \in \mathcal{X}} \langle y, \Theta_{t-1} + \tilde{\theta}_t + \eta_t \gamma \rangle$ *// Update the discrete cache vector*
  $\Theta_t = \Theta_{t-1} + \theta_t$ *// Receive the request for slot t and update total gradient*
end

3.5. Caching through Optimistic Perturbations (OFTPL-Cache)

We propose next a new OFTPL algorithm that significantly improves previous OFTPL proposals [133, 134], 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 [34]. The improvement is possible by setting the perturbation parameters η_t in a manner that is adaptive to prediction error witnessed until $t-1$.

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 ($\mathcal{O}(N)$) time using, e.g., the Median-of-Medians algorithm [137]. The steps of the proposed scheme are presented in Algorithm 5, 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 3.4. *Algorithm 5 ensures, for any time horizon T and $N \geq 2C$ with $C \geq 11$, 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 func-

[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., $f(\lambda a) = \lambda f(a), \lambda > 0$).

tion for each t, is defined as:

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

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:

$$\Phi_t(\Theta_t) = \Phi_t(\Theta_{t-1}+\tilde{\theta}_t) + \langle\nabla\Phi_t(\Theta_{t-1}+\tilde{\theta}_t), \theta_t-\tilde{\theta}_t\rangle + \frac{1}{2}\langle\theta_t-\tilde{\theta}_t, \nabla^2\Phi_t(\hat{\theta}_t)\ (\theta_t-\tilde{\theta}_t)\rangle, \tag{3.6}$$

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 $\Phi_t(\cdot)$:

$$\Phi_t(\Theta_{t-1}+\tilde{\theta}_t) \leq \Phi_t(\Theta_{t-1}) + \langle\nabla\Phi_t(\Theta_{t-1}+\tilde{\theta}_t), \tilde{\theta}_t\rangle. \tag{3.7}$$

From (3.6) and (3.7), we can eventually write:

$$\Phi_t(\Theta_t) \leq \Phi_t(\Theta_{t-1}) + \langle\nabla\Phi_t(\Theta_{t-1}+\tilde{\theta}_t), \theta_t\rangle + \frac{1}{2}\langle\theta_t-\tilde{\theta}_t, \nabla^2\Phi_t(\hat{\theta}_t)(\theta_t-\tilde{\theta}_t)\rangle. \tag{3.8}$$

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

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

where (a) stems from [138, Prop. 2.2]. Thus, (3.8) can be written as:

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

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

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

Then, by Jensen's inequality: $\max_{y\in\mathcal{X}}\mathbb{E}_\gamma\left[\langle y, \Theta_T+\eta_T\gamma\rangle\right] = \max_{y\in\mathcal{X}}\langle y, \Theta_T\rangle = \Phi(\Theta_T) \leq \Phi_T(\Theta_T)$, 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:

$$R_T \leq$$

$$\Phi(\Theta_T) - \sum_{t=1}^{T} \mathop{\mathbb{E}}_{\gamma}[\langle \theta_t, y_t \rangle] \leq \sum_{t=1}^{T} \left(\Phi_t(\Theta_{t-1}) - \Phi_{t-1}(\Theta_{t-1}) + \frac{1}{2}\|\theta_t - \tilde{\theta}_t\|_{H_t} \right). \quad (3.9)$$

We now bound the first term in the RHS of inequality (3.9):

$$\begin{aligned}
&\sum_{t=1}^{T} \Phi_t(\Theta_{t-1}) - \Phi_{t-1}(\Theta_{t-1}) = \sum_{t=1}^{T} \mathop{\mathbb{E}}_{\gamma}[\Phi(\Theta_{t-1} + \eta_t\gamma) - \Phi(\Theta_{t-1} + \eta_{t-1}\gamma)] \\
&\overset{(a)}{\leq} \sum_{t=1}^{T} \mathop{\mathbb{E}}_{\gamma}[\Phi((\eta_t - \eta_{t-1})\gamma)] \overset{(b)}{\leq} \sum_{t=1}^{T} (\eta_t - \eta_{t-1}) \mathop{\mathbb{E}}_{\gamma}[\Phi(\gamma)] \leq \eta_T \mathop{\mathbb{E}}_{\gamma}\left[\max_{y} \langle y, \gamma \rangle\right] \\
&\overset{(c)}{\leq} \eta_T \sqrt{2C \ln \binom{N}{C}} \overset{(d)}{\leq} \eta_T C \sqrt{2 \ln(Ne/C)}, \qquad (3.10)
\end{aligned}$$

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., [34, 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 (3.9). From [34, Eqn. (4)], the $(i,j)^{\text{th}}$ entry of the Hessian matrix is given by $H^t_{i,j} = \frac{1}{\eta_t}\mathbb{E}\big[\tilde{y}(\hat{\theta}_t + \eta_t\gamma)_i\gamma_j\big]$, 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:

$$\begin{aligned}
|H^t_{i,j}| &= \frac{1}{\eta_t} \left| \mathop{\mathbb{E}}_{\gamma}\left[\tilde{y}(\hat{\theta} + \eta_t\gamma)_i\gamma_j\right] \right| \leq \frac{1}{\eta_t} \mathop{\mathbb{E}}_{\gamma}\left[|\tilde{y}(\hat{\theta} + \eta_t\gamma)_i| \ |\gamma_j|\right] \\
&\leq \frac{1}{\eta_t} \mathop{\mathbb{E}}_{\gamma}[|\gamma_i|] \leq \frac{1}{\eta_t}\sqrt{\frac{2}{\pi}}, \qquad (3.11)
\end{aligned}$$

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. (3.9) can be bounded as follows:

$$\begin{aligned}
\|\theta_t - \tilde{\theta}_t\|_{H_t} &= \langle \theta_t - \tilde{\theta}_t, H_t(\theta_t - \tilde{\theta}_t) \rangle = \sum_{i,j} (\theta_{t,i} - \tilde{\theta}_{t,i}) H^t_{ij} (\theta_{t,j} - \tilde{\theta}_{t,j}) \\
&\overset{(a)}{\leq} \sum_{i,j} |(\theta_{t,i} - \tilde{\theta}_{t,i})||H^t_{ij}||(\theta_{t,j} - \tilde{\theta}_{t,j})| \overset{(b)}{\leq} \frac{1}{\eta_t}\sqrt{\frac{2}{\pi}} (\sum_{i} |(\theta_{t,i} - \tilde{\theta}_{t,i})|)^2 \\
&= \frac{1}{\eta_t}\sqrt{\frac{2}{\pi}} ||\theta_t - \tilde{\theta}_t||_1^2. \qquad (3.12)
\end{aligned}$$

where (a) follows from the triangle inequality and (b) from the bound (3.11).

Another way to bound $\|\theta_t - \tilde{\theta}_t\|_{H_t}$, which will be useful later[13], starts from (3.6)

[13] This second bound on the norm enables us to set η_t parameters based solely on the prediction error witnessed so far $\sum_{\tau=1}^{t-1} \|\theta_\tau - \tilde{\theta}_\tau\|$. Consequently, the regret will depend solely on a scaled prediction error (without additive constants).

to get:

$$\frac{1}{2}\|\theta_t - \tilde{\theta}_t\|_{H_t} = \Phi_t(\Theta_t) - \Phi_t(\Theta_{t-1} + \tilde{\theta}_t) - \langle \nabla\Phi_t(\Theta_{t-1} + \tilde{\theta}_t), \theta_t - \tilde{\theta}_t\rangle$$
$$= \mathop{\mathbb{E}}_{\gamma}\left[\Phi(\Theta_t + \eta_t\gamma) - \Phi(\Theta_{t-1} + \tilde{\theta}_t + \eta_t\gamma)\right] + \langle \nabla\Phi_t(\Theta_{t-1} + \tilde{\theta}_t), \tilde{\theta}_t - \theta_t\rangle \overset{(a)}{\leq} \Phi(\theta_t - \tilde{\theta}_t)$$
$$+ \langle \nabla\Phi_t(\Theta_{t-1} + \tilde{\theta}_t), \tilde{\theta}_t - \theta_t\rangle = \max_{y\in\mathcal{X}}\langle y, \theta_t - \tilde{\theta}_t\rangle + \langle \nabla\Phi_t(\Theta_{t-1} + \tilde{\theta}_t), \tilde{\theta}_t - \theta_t\rangle$$
$$\overset{(b)}{\leq} 2\|\theta_t - \tilde{\theta}_t\|_1, \tag{3.13}$$

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 (3.12) and (3.13), we get:

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

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:

$$\frac{1}{2}\|\theta_t - \tilde{\theta}_t\|_{H_t} \leq \min\left(\frac{||\theta_t - \tilde{\theta}_t||_1^2}{\sqrt{2\pi}\beta\sqrt{\sum_{\tau=1}^{t-1}||\theta_\tau - \tilde{\theta}_\tau||_1^2}}, 2\frac{||\theta_t - \tilde{\theta}_t||_1^2}{\sqrt{2\pi}\beta\sqrt{||\theta_t - \tilde{\theta}_t||_1^2}}\right)$$
$$\overset{(a)}{\leq} \frac{3}{\sqrt{2\pi}\beta}\frac{||\theta_t - \tilde{\theta}_t||_1^2}{\sqrt{\sum_{\tau=1}^{t}||\theta_\tau - \tilde{\theta}_\tau||_1^2}} \tag{3.14}$$

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 (3.10), and the per-step regret bounds (3.14), we can substitute them in (3.9) to get:

$$\mathop{\mathbb{E}}_{\gamma}[R_T] \leq \eta_T C\sqrt{2\log(Ne/C)} + \frac{3}{\sqrt{2\pi}\beta}\sum_{t=1}^{T}\frac{||\theta_t - \tilde{\theta}_t||_1^2}{\sqrt{\sum_{\tau=1}^{t}||\theta_\tau - \tilde{\theta}_\tau||_1^2}}. \tag{3.15}$$

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 (3.15) and using the definition of η_T we get the regret upper bound:

$$\mathbb{E}_\gamma[R_T] \leq \sqrt{\sum_{\tau=1}^{T} ||\theta_\tau - \tilde{\theta}_\tau||_1^2} \left(C\sqrt{2\ln(Ne/C)})\beta + \frac{6}{\beta\sqrt{2\pi}} \right) \tag{3.16}$$

Optimizing over the constant β, we get that $\beta = \sqrt{\frac{3}{C}}(\frac{1}{\pi \ln(Ne/C)})^{1/4}$, $C \geq 11$ (Recall we that $0 < \beta \leq 1/\sqrt{2\pi}$). Substituting this valud for β back in (3.16) we arrive at the result. □

Discussion. Similarly to Theorem 3.2, the regret bound here is modulated with the quality of predictions: it collapses to zero when predictions are perfect, and maintains $R_T \leq 3.68 \ \ln\left(\frac{Ne}{C}\right)^{1/4} \sqrt{2CT}$ 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 [58] 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 3.2, OFTPL achieves regret bounds worse by a factor of $\sim 1.9 \ln\left(\frac{Ne}{C}\right)^{1/4}$, 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 5 does not involve the expensive projection operation that appears in Algorithm 4, but rather a simple quantile-finding operation (top C files) with a worst-case complexity of $\mathcal{O}(N)$. 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 3.4 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[14][58, 64, 65]. It is also interesting to note that the bounds of Theorem 3.4, and 3.2, 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. Lastly, we note that we can sample a fresh perturbation vector at each time step in order to handle adaptive adversaries (i.e., adversaries that do not fix the cost sequence in advance but react to the choices of the algorithm). The same analysis applies since perturbations are equal in distribution. We refer the reader to e.g., [139, Sec. 8] for techniques for reducing guarantees from oblivious to adaptive adversaries via fresh-sampling.

[14]The same property of depending on prediction mistakes rather than T holds for Theorem 3.2 but the regret scales as the square root of double the number of mistakes, due to the use of ℓ_2 norm.

3.6. 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 $x^\star = y^\star \triangleq \arg\max_{x \in \mathcal{X}_s} \langle x, \Theta_T \rangle$. We present two solution approaches for this problem, using both OFTRL and OFTPL. These results are of independent interest with applications beyond caching.

3.6.1. Approximate OFTPL

Similarly to Algorithm 5, 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 [140], 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. 3.5. 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 [128], 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 [141]:

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

for a positive constant α. This generalized regret metric allows to use a parameterized benchmark, in line with prior works, e.g., see [65] and references therein. Now, it is important to see that while the Knapsack problem admits an FPTAS [142], 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 [141]. 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:

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

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 6. 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 $1/2$; Set $S = \{k\}$ with probability $1/2$.

Algorithm 6: `OFTPL-UneqCache`

1 **Input**: $\eta_1 = 0$, $y_1 = \arg\min_{y \in \mathcal{X}_s} \langle y, \theta_1 \rangle$, $s = (s_i, i \in \mathcal{N})$
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*
end

2. Set $y_i^t = 0, \forall i \in \mathcal{N}$; and update to $y_i^t = 1$ 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 characterizes the guarantees of Algorithm 6.

Theorem 3.5. *Algorithm 6 ensures, for any time horizon T, the expected regret bound:*

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

Discussion. The bounds of Theorem 6 possess 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 1/2 the utility of the best-in-hindsight utility $\langle \Theta_T, y^\star \rangle$. Relying upon such approximations is an inevitable concession for NP-hard problems – especially when having to solve them repeatedly. Interestingly, the computational complexity of Algorithm 6 is comparable to that of Algorithm 5, 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.

Proof of Theorem 3.5. 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

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

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

$$R_T^{(1/2)} = \frac{1}{2}\sum_{t=1}^{T}\langle \theta_t, y^\star\rangle - \mathbb{E}\left[\sum_{t=1}^{T}\langle \theta_t, y_t\rangle\right] \overset{(a)}{\leq} \frac{1}{2}\left(\sum_{t=1}^{T}\langle \theta_t, y^\star\rangle - \sum_{t=1}^{T}\langle \theta_t, \hat{y}_t\rangle\right)$$
$$\overset{(b)}{=} 1.84\sqrt{C}\left(\ln\frac{Ne}{C}\right)^{1/4}\sqrt{\sum_{t=1}^{T}\|\theta_t - \tilde{\theta}_t\|_1^2}.$$

where inequality (a) follows from the $1/2$-approximation property of the randomized rounding algorithm (3.18); and (b) follows from the result of Theorem[15] 3.4.

□

3.6.2. 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. 3.6.1, we cannot apply the $(1/2)$-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 [127], which is known to achieve the useful property re-stated below.

Lemma 3.6. *For $a \in [0,1]^N$, the `DepRound` scheme in Algorithm 14 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 $(1/2)$-Regret guarantee w.r.t. the same benchmark as in Sec. 3.6.1. The steps are presented in Algorithm 7. The diligent reader will observe that essentially we merge steps from Algorithm 4 and Algorithms 6. The detailed steps of the `DepRound` subroutine are presented in the Appendix. The next theorem characterizes the algorithm's performance.

Theorem 3.7. *Algorithm 7 ensures, for any time horizon T, the expected regret bound:*

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

[15] Theorem 3.4 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 5 ($\{0,1\}$ decision variables with non-negative coefficients).

Algorithm 7: `OFTRL-UneqCache`

Input: $\sigma = 1/\sqrt{C}$, $\delta_1 = \|\theta_1 - \tilde{\theta}_1\|_2^2$, $\sigma_1 = \sigma\sqrt{\delta_1}$, $x_1 = \arg\min_{x\in\mathcal{X}_s}\langle x, \theta_1\rangle$
Output: $\{y_t \in \mathcal{X}_s\}_T$ *// Feasible discrete caching vector at each slot*
for $t = 2, 3, \ldots$ **do**
  $\tilde{\theta}_t \leftarrow$ `prediction` *//Obtain gradient prediction for slot t*
  $\hat{x}_t = \arg\max_{x\in\mathrm{conv}(\mathcal{X}_s)} \left\{-r_{1:t-1}(x) + \langle x, \Theta_{t-1} + \tilde{\theta}_t\rangle\right\}$
  $\bar{x}_t \longleftarrow$ `DepRound`$(\hat{x}_t)$ *// Compute the "almost integral" cache vector*
  $x_t \longleftarrow$ `Rand`$(\bar{x}_t, k)$ *// Perform Randomized Rounding*
  $\Theta_t = \Theta_{t-1} + \theta_t$ *// Receive t-slot request and update total grad*
  $\sigma_t = \sigma\left(\sqrt{\delta_{1:t}} - \sqrt{\delta_{1:t-1}}\right)$ *// 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 7, 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 7, 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.

Proof of Theorem 3.7. First, we show that the $1/2$-point-wise approximation holds for $\{x_t\}_t$. Then, we re-use the result of Theorem 3.2. In detail, by Lemma 3.6, the `DepRound` subroutine 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 3.5, we have that $\mathbb{E}[x_t] \geq \frac{1}{2}\bar{x}_t$, where $x_t \in \mathcal{X}_s$. We recover our $1/2$-approximation guarantee for OFTRL iterates $\{x_t\}_t$:

$$\mathbb{E}[x_t] \geq \frac{1}{2}\hat{x}_t. \tag{3.19}$$

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

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

where inequality (a) follows from the $1/2$-approximation property of the randomized rounding algorithm (3.19); and (b) follows from the result of Theorem 3.2. Although

that theorem states the bound for the regret of the integral decisions $\{x_t\}_t$, we have seen in its proof that the bound is essentially the same for the continuous actions. □

3.7. 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 $d = \big(d_{ij} \in \{0, 1\} : i \in \mathcal{I}, j \in \mathcal{J}\big)$, 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 [30, 43].

We introduce the new caching variables k and routing variables u. Namely, $k^t_{nj} \in \{0, 1\}$ 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 $k_t = (k^t_{nj} : n \in \mathcal{N}, j \in \mathcal{J})$ belongs to

$$\mathcal{K} = \left\{ k \in \{0, 1\}^{N \cdot J} \;\middle|\; \sum_{n \in \mathcal{N}} k_{nj} \leq C_j, \; j \in \mathcal{J} \right\},$$

where C_j is the capacity of cache $j \in \mathcal{J}$. We use the routing variable $u^t_{nij} \in \{0, 1\}$ to decide the delivery of file n to user i from cache j, and define the t-slot routing vector $u_t = (u^t_{nij} \in [0, 1] : n \in \mathcal{N}, i \in \mathcal{I}, j \in \mathcal{J})$ that is selected from the set:

$$\mathcal{U} = \left\{ u \in \{0, 1\}^{N \cdot J \cdot I} \;\middle|\; \sum_{j \in \mathcal{J}} u_{nij} \leq 1, \; n \in \mathcal{N}, i \in \mathcal{I} \right\}.$$

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 [30]. The request vector θ_t is redefined to reflect a request's source and destination: $\theta_t = (\theta^t_{ni} \in \{0, 1\} : n \in \mathcal{N}, i \in \mathcal{I})$, and is drawn from the set: $\mathcal{Q} = \left\{\theta \in \{0, 1\}^{N \cdot I} \;\middle|\; \sum_{n \in \mathcal{N}} \sum_{i \in \mathcal{I}} \theta_{ni} = 1\right\}$.

We can now introduce the t-slot utility function:

$$f_t(x_t) = \sum_{n \in \mathcal{N}} \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{J}} \theta^t_{ni} k^t_{nij} \; .$$

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 $\mathcal{X}_b \triangleq \{\{\mathcal{K} \times \mathcal{U}\} \cap \{u_{nij} \leq k_{nj} d_{ij}\}\}$. $\mathbb{P}_2$ is known to be NP-Hard via a reduction to the set cover problem [30, Sec. 3],

Algorithm 8: `OFTRL-BipCache`

Input: $\sigma = 1/\sqrt{1+JC}$, $\delta_1 = \|\theta_1 - \tilde{\theta}_1\|_2^2$, $\sigma_1 = \sigma\sqrt{\delta_1}$, $x_1 = \arg\min_{x\in\mathcal{X}_b}\langle x, \theta_1\rangle$
Output: $\{x_t = (k_t, u_t) \in \mathcal{X}_b\}_T$ *// Feasible discrete caching/routing vector at each slot*
for $t = 2, 3, \ldots$ **do**
  $\tilde{\theta}_t \leftarrow$ `prediction` *//Obtain request prediction for slot t*
  $\hat{x}_t = \arg\max_{x\in \mathrm{conv}(\mathcal{X}_b)} \left\{-r_{1:t-1}(x) + \langle x, \Theta_{t-1} + \tilde{\theta}_t\rangle\right\}$ *// Update the continuous policy vector*
  $(k_j^t) \longleftarrow MadowSample(\hat{k}_t), \forall j \in \mathcal{J}$ *// Obtain the discrete cache vector for each cache independently*
  $\Theta_t = \Theta_{t-1} + \theta_t$ *// Receive t-slot request and update total grad*
  Set $u_{nij'}^t \leftarrow 1$ for a randomly selected $j' \in \mathcal{J}^{ni}$ *//Update the discrete routing vector*
  $\sigma_t = \sigma\left(\sqrt{\delta_{1:t}} - \sqrt{\delta_{1:t-1}}\right)$ *// Update the regularization parameter*
end

[65, 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[16]. The detailed steps of the proposed OFTRL schemed are presented in Algorithm 8, where we reuse the regularization scheme from Sec. 3.4 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.

Theorem 3.8. *Algorithm 8 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 3.8 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 [65][17]. This improvement comes at the expense of a projection operation in the OFTRL step. Additionally, Algorithm 8

[16] In other words, the routing variables are auxiliary in the Femtocaching model, and indeed in [30] these variables where omitted, while they appear with different name in subsequent works, e.g., as virtual caching variables in [65].

[17] We note that the bound in [65] 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.

manages to reduce further the constant terms when it has access to high quality predictions for the next-slot requests.

Proof. We start from the result of [75, Thm. 1] (or its earlier version from [123]), 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 $\mathbb{E}[\hat{k}] = k$, $\mathbb{E}[\hat{u}] \geq (1 - 1/e)u$, 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:

$$R_T^{(1-1/e)} = (1 - \frac{1}{e}) \sum_{t=1}^{T} \langle \theta_t, \hat{x}^\star \rangle - \mathbb{E} \sum_{t=1}^{T} \langle \theta_t, x_t \rangle \overset{(a)}{\leq} 1 - \frac{1}{e} \big(\sum_{t=1}^{T} \langle \theta_t, \hat{x}^\star \rangle - \sum_{t=1}^{T} \langle \theta_t, \hat{x}_t \rangle \big)$$
$$\overset{(b)}{=} (2 - \frac{2}{e}) \sqrt{1 + JC} \sqrt{\sum_{t=1}^{T} \|\theta_t - \tilde{\theta}_t\|_2^2}.$$

where inequality (a) follows from the $(1 - 1/e)$-approximation property of the randomized rounding algorithm; and (b) follows from the result of Theorem [75, Thm. 1].

Now, to show that $\mathbb{E}[\hat{k}] = k$, we follow the same argument in the proof of Theorem 3.2. 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, $\mathbb{E}[k_{nj}] = \hat{k}_{nj}, \forall n, j$. 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

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

Where in the above chain of inequalities, (a) follows from u_{nij} being 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 $[0, 1]$.

□

3.8. 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 [62] 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 [75, 123]. 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:

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

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 $z^\star = \arg\max_{z\in\text{conv}(\mathcal{X})}\langle \Theta_T, z\rangle$. 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 $\Delta_2 = \left\{w \in [0,1]^2 \middle| \|w\|_1 = 1\right\}$, 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:

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

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 [123, Thm. 3]:

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

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

$$\mathbb{E}\left[R_T\{z_t\}\right] \leq R_T^{(w)} + \min\left\{R_T^{(p)}, R_T^{(o)}\right\}. \tag{3.24}$$

The steps of this scheme are summarized in Algorithm 9.

Algorithm 9: `Experts-Cache`

Input: $z_1 \in \mathcal{X}$
Output: $\{z_t \in \mathcal{X}\}_T$ // *Feasible caching vector at each slot*
for $t = 2, 3, \dots$ **do**
  $\hat{z}_t^{(p)} = \mathcal{P}_{\text{conv}(\mathcal{X})}\left\{z_{t-1}^{(p)} + \frac{1}{\sqrt{t}}\theta_t\right\}$ // *Pessimistic expert makes proposal*
  $z^{(o)} = \arg\max_{z \in \mathcal{X}} \langle \tilde{\theta}_t, z \rangle$ // *Optimistic expert makes proposal based on the oracle's prediction*
  $\hat{z}_t = w_t^{(p)} \hat{z}_t^{(p)} + w_t^{(o)} z_t^{(o)}$ // *Meta-learner combines proposals*
  $z_t \leftarrow MadowSample(\hat{z}_t)$ // *Obtain the discrete cache vector using Algorithm 13*
  $\Theta_t = \Theta_{t-1} + \theta_t$ // *Receive the request for slot t and update total grad*
  $\hat{w}_t = \mathcal{P}_{\text{conv}(\Delta_2)}\left\{w_{t-1} + \frac{1}{\sqrt{t}} l_t\right\}$ // *Meta-learner observes losses l_t & updates weights*
end

Discussion. The performance advantage of the bound in (3.24) 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 $\mathcal{O}(\sqrt{T})$ 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 [62] for that operation. It is also important to stress that this framework allows to combine more than one expert, in order to either to e.g., include more than one predictor, see discussion also in [75].

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 [143], and evaluated in several cases, e.g., see [144] and reference therein, which however do not consider predictors nor provide any theoretical analysis (or, bounds) for the performance of this approach.

3.9. Comparative Summary & Experiments

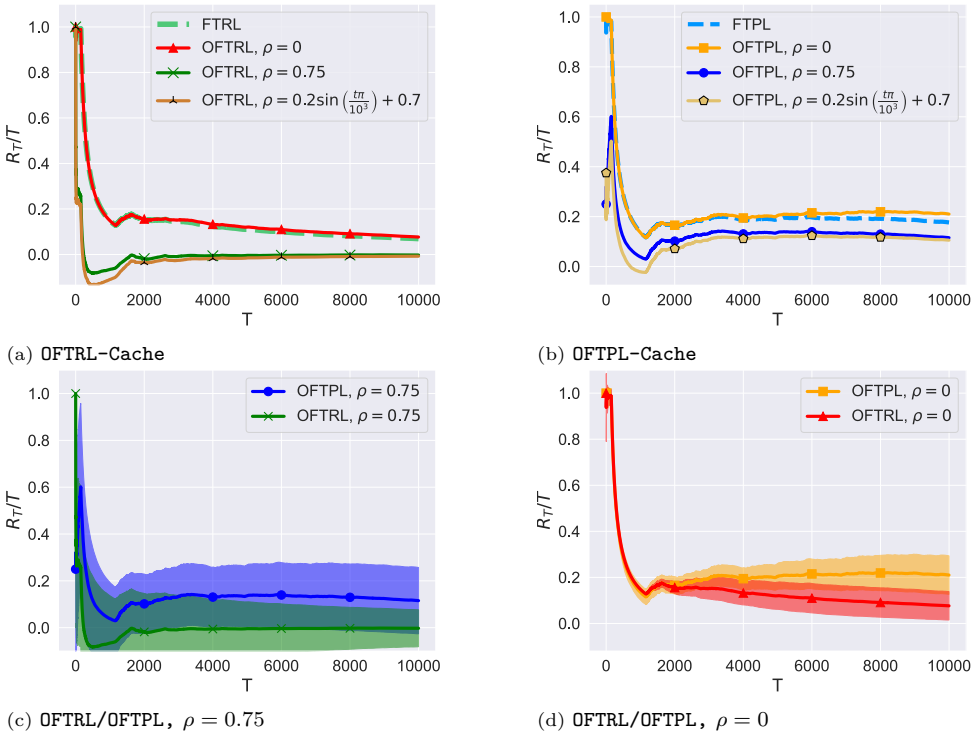
Table 3.1 shows the performance and complexity trade-offs for the presented algorithms, and compares them to recent studies of discrete no-regret caching in the literature. The best case refers to the situation where the request predictions are perfect $\tilde{\theta}_t = \theta_t, \forall t$. The worst 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 [58] and [64] assume and utilize knowledge of the time horizon

T ([66] 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 [112], which argues about the advantages of adaptive algorithms of the sort presented here. While the authors in [58] discuss the bipartite model, their simpler linear *elastic* model of utility is different than the one considered here (see [58, 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 $R_T^{(w)} \leq \mathcal{O}(\sqrt{T})$ 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.

Table 3.1: 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 (3.17)).

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	$\mathcal{O}(\sqrt{T})$	$\mathcal{O}(N^2)$	1	✓
2	• Single cache • Predictions	0	$\mathcal{O}(\textit{poly-log}(N)\sqrt{T})$	$\mathcal{O}(N)$	1	✓
3	• Single cache • Predictions • Unequal sizes	0	$\mathcal{O}(\sqrt{T})$	$\mathcal{O}(N^2)$	$1/2$	✓
4	• Single cache • Predictions • Unequal sizes	0	$\mathcal{O}(\textit{poly-log}(N)\sqrt{T})$	$\mathcal{O}(N)$	$1/2$	✓
5	• Bipartite Network • Predictions	0	$\mathcal{O}(\sqrt{T})$	$\mathcal{O}(N^2)$	$1 - 1/e$	✓
6	• Single Cache • Predictions	$b < 0$	$R_T^{(w)} + \mathcal{O}(\sqrt{T})$	$\mathcal{O}(N)$	1	✓
[58]	• Single cache	$\mathcal{O}(\textit{poly-log}(N)\sqrt{T})$		$\mathcal{O}(N)$	1	–
[64]	• Single cache	$\mathcal{O}\left(\sqrt{T}\right)$		$\mathcal{O}(N)$	1	–
[65]	• Bipartite network	$\mathcal{O}(\textit{poly-log}(N)\sqrt{T})$		$\mathcal{O}(N)$	$1 - 1/e$	✓
[66]	• General network	$\mathcal{O}(\textit{poly-log}(N)\sqrt{T})$		$\mathcal{O}(N)$	$1 - 1/e$	–

Experiments. We compare the performance of our algorithms with carefully-selected competitors: the FTRL policy, which generalizes the OGD from [62], and the FTPL method from [58]. The full codebase for the proposed policies and experiments is available via GitHub [145]. 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 [105], 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



(a) `OFTRL-Cache`

(b) `OFTPL-Cache`

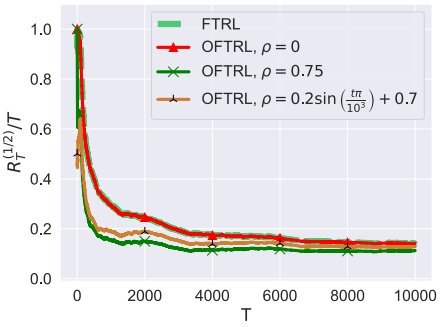
(c) `OFTRL/OFTPL`, $\rho = 0.75$

(d) `OFTRL/OFTPL`, $\rho = 0$

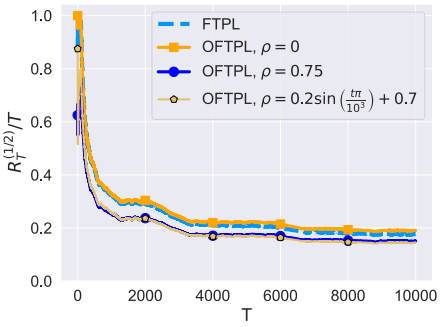
Figure 3.2: Comparison of R_T/T for **equal-sized** files in a **single cache** and different policies: (a) `OFTRL-Cache` vs. FTRL/OGD; (b) `OFTPL-Cache` vs. FTPL [58]; (c) `OFTRL-Cache` vs. OFTPL-Cache for good predictions and in (d) for worst-case predictions. In (c), (d) we plot the 0.95-confidence interval (8 runs).

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 3.9). 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.

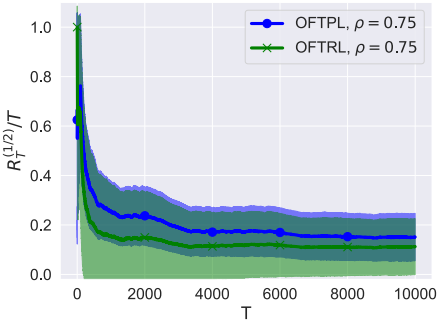
Fig. 3.2 shows the average regret (hit-rate gap to the optimal) growth with time for FTPL [58], FTRL [62], and their proposed optimistic counterparts. We experiment with $\rho = 0$, and $\rho = 0.75$. If, e.g., the request predictions were based on recommendations, these reflect the cases where the users do not follow the recommendations ($\rho = 0$), or actually request the recommended movie/file with probability 75%, ($\rho = 0.75$). In addition, we experiment with a sinusoidal ρ, which varies



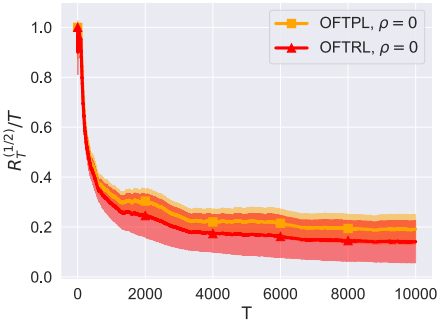
(a) `OFTRL-UneqCache`



(b) `OFTPL-UneqCache`



(c) `OFTRL/OFTPL`, $\rho = 0.75$



(d) `OFTRL/OFTPL`, $\rho = 0$

Figure 3.3: Comparing $R_T^{(1/2)}/T$ for **unequal-sized** files, **single cache**. (a) `OFTRL-UneqCache` vs. FTRL/OGD; (b) `OFTPL-UneqCache` vs. FTPL [58]; (c)-(d) `OFTRL-UneqCache` vs. OFTPL-UneqCache for good (bad) predictions.

between $\rho = 0.5$ and $\rho = 0.9$, with a period of 10^3 slots. We observe that optimism accelerates and improves learning the best files to cache, reaching an average improvement of 104% (for OFTRL) and 37.1% (for OFTPL) when $\rho = 0.75$, compared to their "vanilla" counterparts (no predictions). Moreover, the performance degradation due to inaccurate predictions is almost negligible: $\leq 8.3\%$ for OFTRL and $\leq 6.6\%$ for OFTPL. We also plot the 0.95-confidence interval of R_T in Figures 3.2c and 3.2d, where we note the more condensed distribution for OFTRL: 44.3% and 26.1% tighter at $t = 10k$ when $\rho = 0.75, \rho = 0$, respectively. This is because the distribution ($\{\hat{x}_t\}_T$ iterates in OFTRL) becomes 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. 3.3 we evaluate the algorithms for the *unequal* sizes case and plot the $1/2$-regret. We observe the same pattern of negligible performance degradation when $\rho = 0$, while $\rho = 0.75$ enables an improvement of 35% (for OFTRL) and 18.8% (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.

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. 3.4 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 [104] with $N = 10^4$ and $C = 150$. R_{5k} is measured using prediction vectors with varying density that is placed on the file to be requested. Namely, if at step t, the requested file is n, we feed the optimistic algorithms with a prediction vector:

$$\tilde{\theta} \quad \text{with:} \quad \tilde{\theta}^t_n = \zeta \quad \text{and} \quad \tilde{\theta}^t_{n'} = (1-\zeta)/(N-1), \ \forall n' \neq n.$$

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 9.8% and 1.4%, 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 59.6% compared to OFTPL, when fed the same predictions, at the expense of its additional computation complexity.

Algorithms 8 and 9

Fig. 3.5 plots the regret for the expert-based Algorithm 9. 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 (3.24) 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 $R_T = 0.113$ at time $T = 10k$ compared to $R_T = 0.075$ (OFTRL). Lastly, the bipartite *utility* is shown in Fig. 3.6, 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 $T = 10k$.

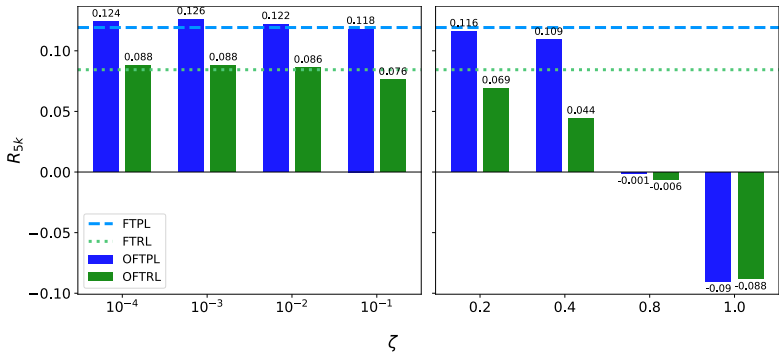


Figure 3.4: Regret with varying probability mass placed on the correct file in the prediction vector.

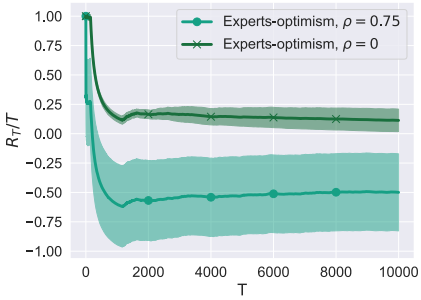


Figure 3.5: Average regret of experts-based optimism.

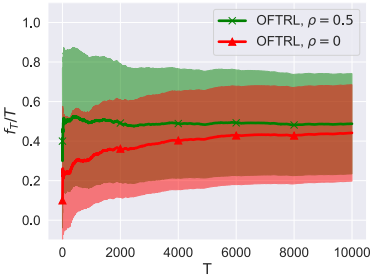


Figure 3.6: Average OFTRL hit-rate for cache network.

3.10. Conclusion

In this chapter, 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 $\mathcal{O}(\sqrt{T})$.

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 [58], 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 [146] and the minimum regret over all finite-state-predictors [147], and extending the study towards such more-refined benchmarks is certainly interesting. In fact, the following chapter will outline a

general method that can be used to extend all OFTRL variants, including the ones discussed in this chapter, into the more refined *dynamic regret* metric, where the optimal cache configuration can itself change over time. Finally, considering unequal routing utility (e.g., link-capacitated model [148]) and unequal-sized files for the bipartite network model remains an open question [149].

4

Optimism Under a Universal Metric

In the previous two chapters, we evaluated the performance of our learning algorithms using the *static* regret metric, which measures performance relative to the best-fixed decision in hindsight. While static regret provides strong guarantees in stationary environments, it fails to capture how well an algorithm adapts when the optimal decision evolves over time, such as in response to shifting user preferences. In many real-world decision-making problems, a fixed benchmark is overly restrictive, motivating the need for a more general performance measure. To address this, we shift our focus in this chapter to *dynamic* regret, which benchmarks an algorithm against a sequence of optimal decisions that vary over time. However, achieving tight dynamic regret bounds introduces new challenges. Prior work has suggested that the FTRL algorithmic family, which has been our main tool so far, struggles in dynamic environments due to its tendency to consider all past costs in producing the actions, making it difficult to adapt to changing conditions.

In this chapter, we revisit FTRL, and its optimistic variant, showing that by selectively refining past information to prevent excessive reliance on redundant data, an idea we call pruning, we can indeed achieve dynamic regret bounds. More broadly, our analysis sheds light on the important role of optimism in highly dynamic settings, demonstrating how it can enhance adaptability without compromising robustness. Formally, we **redefine** the regret $\mathcal{R}_T$ in this chapter to be with respect

The content of this chapter has been published in

- Naram Mhaisen and George Iosifidis. "On the Dynamic Regret of Following the Regularized Leader: Optimism with History Pruning". In *Proceedings of the International Conference on Machine Learning (ICML)*. 2025.

to a possible changing comparator :

$$\mathcal{R}_T \doteq \sum_{t=1}^{T} f_t(\boldsymbol{x}_t) - f_t(\boldsymbol{u}_t),$$

where $\{\boldsymbol{u}_t\}_{t=1}^T \in \mathcal{X}$ is *any* set of comparators with desirable costs that we wish to benchmark against. The dynamic regret is thus simply the performance gap between the learner and the comparator sequence. A key complexity measure associated with this sequence is its path length P_T, given by

$$P_T \doteq \sum_{t=1}^{T-1} \|\boldsymbol{u}_{t+1} - \boldsymbol{u}_t\|. \tag{4.1}$$

The path length quantifies the variation in the comparator sequence, with larger values indicating a more dynamic environment and a more challenging learning setting.

4.1. Background and Motivation

In this section, we briefly recap the concepts of data dependence and optimism in online learning, introduce useful technical definitions, and then transition to discussing their special importance for the dynamic regret metric. As indicated throughout this thesis, algorithms with a sub-linear regret guarantee have been behind recent state-of-art advances not only in classical computer science problems such as caching [150], portfolio management [151], and generalized assignment [152], but also in machine learning problems such as the design of (enhanced) ADAM optimizer [153], sub-modular optimization [154], and supervised learning with shifting labels [155] among others. However, even in the *static* settings where the comparator is fixed: $\boldsymbol{u}_t = \boldsymbol{u}, \forall t$, the strategy of minimizing witnessed costs at each t admits *linear* regret, indicating a failure in learning [156, Sec 2.2]. Hence, careful *regularization* is needed to avoid overfitting past data, which is the fundamental idea behind the two main algorithmic families for OCO: Follow the Regularized Leader (FTRL) and Online Mirror Descent (OMD). It is known that both frameworks achieve an order optimal static regret bound of $\mathcal{O}(\sqrt{T})$ [28, Sec. 5]. Further, it has been shown that when the regularization is made *data-dependent*, the regret bounds will indeed be sub-linear and, more importantly, data-dependent. This is a desirable, albeit challenging, objective.

Data-dependent bounds are preferable because they are parametrized by the actual problem instance, $\{f_t(\cdot)\}_{t=1}^T$, rather than crude universal bounds on this instance. I.e., on the Lipschitz constant L and the horizon T. While this complicates the analysis, it leads to more custom algorithms in which the "easiness" of an instance is reflected in the bound. For example, AdaGrad-style bounds [157], and follow-ups, achieve static regret of the form $\mathcal{O}(\sqrt{G_T})$ where $G_T = \sum_{t=1}^T \|\boldsymbol{g}_t\|^2$, $\boldsymbol{g}_t \in \partial f_t(\boldsymbol{x}_t)$ is the sub-gradients norm trajectory. This form of bounds is desirable since they scale with the lengths $\|\boldsymbol{g}_t\|$ and hold for any value T rather than being dependent on the worst case L value and on a single pre-provided T. Yet, they

maintain the order-optimal bound ($\mathcal{O}(\sqrt{T})$) in all cases. Similarly, [158] achieves $\mathcal{O}(\sqrt{V_T})$ for smooth functions where $V_T \doteq \sum_{t=2}^{T} \max_{\boldsymbol{x}} \|\nabla f_t(\boldsymbol{x}) - \nabla f_{t-1}(\boldsymbol{x})\|^2$ is the gradient variation trajectory, which, again, is never more $\mathcal{O}(\sqrt{T})$ but tighter for slowly-varying functions.

A more general problem-dependent quantity is the accumulated *prediction error* [26, 125]; suppose the learner receives a prediction $\tilde{f}_t(\cdot)$ for the cost function $f_t(\cdot)$ *prior* to deciding $\boldsymbol{x}_t$, with no guarantees on its accuracy, the quantity of interest is:

$$E_T \doteq \sum_{t=1}^{T} \epsilon_t^2, \quad \epsilon_t \doteq \|\boldsymbol{g}_t - \tilde{\boldsymbol{g}}_{\boldsymbol{t}}\|, \tag{4.2}$$

where $\tilde{\boldsymbol{g}}_t \in \partial \tilde{f}_t(\boldsymbol{x}_t)$. Clearly, we can choose $\tilde{f}_t(\cdot)$ to be 0 or $f_{t-1}(\cdot)$, recovering the dependence on G_T and V_T[1], respectively. Algorithms whose regret depends on E_T are called *optimistic* and are crucial for achieving best-of-both-worlds-style guarantees: *constant* regret in predictable environments and sub-linear in all cases, delivering adaptability without sacrificing robustness. Interestingly, the application of optimistic algorithms extends beyond enabling the use of untrusted predictions in the OCO problem; they have been shown to be key in the more general *delayed* OCO problem [86], as well as the related OCO with memory problem [159]. Given their significance, we focus on developing "optimistic" algorithms in this work. That is, algorithms that receive and use predictions of future costs and have regret bounds parametrized by E_T.

While (optimistic) data dependence is well understood for both OMD and FTRL frameworks under the *static* regret metric, the story is different when it comes to dynamic regret. For OMD, the current prevalent form of optimistic problem-dependent dynamic regret bounds first appeared in [160], who used a variant of the Optimistic OMD (OOMD) (two-step variant). This formulation requires that $\boldsymbol{g}_t$ is defined before calculating $\boldsymbol{x}_t$, which is only possible for linear functions (recall $\boldsymbol{g}_t \in \partial f_t(\boldsymbol{x}_t)$). This "cyclic dependency" issue was only addressed recently in [161], who obtained bounds that depend on a quantity similar to E_T, $D_T \doteq \sum_{t=1}^{T} \|\nabla f_t(\boldsymbol{y}_{t-1}) - \nabla \tilde{f}_t(\boldsymbol{y}_{t-1})\|$, where $\boldsymbol{y}_t$ are points generated by an online algorithm.

As for FTRL, the first dynamic regret guarantee has been established by [153], showing $\mathcal{R}_T = \mathcal{O}(P^{1/3}T^{2/3})$ for bounded domains. While this guarantee is not data-dependent and suboptimal in T, it suffices for the authors' goal of explaining the behavior of the Adam optimizer. For bounded domains, to our knowledge, no prior work has established dynamic regret guarantees for FTRL, problem-dependent or not, with $\mathcal{O}(P_T^{\beta}\sqrt{T})$ dependence, for any $\beta \in [0, 1]$. This gap raises an intriguing question regarding the performance of FTRL under the dynamic regret metric, particularly given that FTRL can be *equivalent* to OMD under specific regularization and linearization choices [112, Sec. 6], suggesting its potential applicability in dynamic environments. However, the extent to which FTRL admits meaningful dynamic regret guarantees remains an open problem.

[1] More precisely, for differentiable functions, we recover $V_T' = \sum_{t=1}^{T} \|\nabla f_t(\boldsymbol{x}_t) - \nabla f_{t-1}(\boldsymbol{x}_{t-1})\|^2 \leq V_T$.

Conceptually, the versatility of FTRL arises from its *richer* state representation, where the "state" refers to the vector used to determine the next iterate, $\boldsymbol{x}_{t+1}$. In OMD, the state of the algorithm is merely the current feasible point, $\boldsymbol{x}_t$. In contrast, the state in FTRL is some mapping of all previous cost functions. In the most common case, this mapping is simply the cumulative gradient, $\boldsymbol{g}_{1:t} \doteq \sum_{\tau=1}^{t} \boldsymbol{g}_\tau$. This same versatility, which stems from retaining all past costs, introduces a key drawback: retaining all past costs can hinder adaptation when the costs are nonstationary.

Specifically, it has been demonstrated that FTRL iterates that use such mapping *fail* to achieve sublinear dynamic regret even for *constant* path lengths [162, Thm. 2]. In essence, the aggregation of past costs obscures the switching patterns in the data (i.e., in $\{f_\tau(\cdot)\}_{\tau=1}^{t}$), which are crucial for the iterates to adapt appropriately, particularly when competing with moving comparators. Since most FTRL variants in the literature aggregate past gradients, FTRL is often considered unsuitable for dynamic environments [163]. However, these findings do not dismiss the potential of *all* FTRL variants. On the contrary, they give insight into how to design variants that adapt to changing comparators, a key motivation behind the pruning mechanism we analyze here.

This chapter seeks to address the ambiguity regarding FTRL's performance under dynamic regret. Clarifying this issue is not only intellectually compelling but also important for establishing more refined problem-dependent bounds, as we demonstrate in the sequel. It will also help explain the notable performance gap between lazy (i.e., typically FTRL) and greedy (i.e., typically OMD) methods in dynamic settings. Furthermore, we introduce a new set of FTRL-native analysis tools, expanding its applicability in dynamic environments and paving the way for dynamic regret guarantees in other OCO settings (e.g., delayed feedback or memory constraints), where FTRL is often the framework of choice.

Chapter Notation. We use calligraphic capital letters, e.g., $\mathcal{X}$ to denote sets. Vectors are denoted with bold-face small letters, e.g., $\boldsymbol{a}$, and we use the subscript t to highlight a vector's dependence on a specific time slot e.g., $\boldsymbol{a}_t$. Scalars are denoted with regular letters and can as well depend on the time, e.g., h_t. We denote with $\{\boldsymbol{a}_t\}_{t=1}^{T}$ the sequence of vectors or parameters from slot $t = 1$ up to slot T. Sometimes we use the shorthand $\{\boldsymbol{a}_t\}_T$ for the same. Whenever the horizon is not relevant we use $\{\boldsymbol{a}_t\}_t$. We also use the shorthand sum notation for scalars $b_{1:t} = \sum_{i=1}^{T} b_i$ and the element-wise sum of vectors $\boldsymbol{a}_{1:t} = \sum_{i=1}^{T} \boldsymbol{a}_i$. We denote with $[T]$ the integer set $1, 2, \cdots, T$. We make use of the indicator function $I_{\mathcal{X}}(\boldsymbol{x})$, which evaluates to $I_{\mathcal{X}}(\boldsymbol{x}) = 0$ if $x \in \mathcal{X}$ and ∞ otherwise.

4.2. Methodology and Contributions

As noted earlier, certain forms of FTRL are equivalent to OMD, suggesting that dynamic regret guarantees should hold in these cases. The equivalence arises when the update minimizes the regularized linearized history, which is the starting point of this paper. Specifically, let $r_t(\cdot)$ be a data-dependent strongly convex regularizer that evolves with t, potentially depending on past costs and actions. Then, the

standard linearized FTRL update is

$$\boldsymbol{x}_{t+1} = \arg\min_{\boldsymbol{x}\in\mathcal{X}} \langle \boldsymbol{g}_{1:t}, \boldsymbol{x}\rangle + r_{1:t}(\boldsymbol{x}). \tag{4.3}$$

If $\mathcal{X} = \mathbb{R}^n$, and $r_t(\boldsymbol{x})$ is proximal[2], this update is equivalent to the OMD update which uses $r_{1:t}(\cdot)$ as the mirror map. This is proven in [112, Thm. 11], even for the more general case of composite costs.

Here, we consider the *optimistic* version of this update. That is, we append the prediction $\tilde{f}_{t+1}(\boldsymbol{x})$ to the sum in (4.3). This will later allow us to have problem-dependent bounds that are modulated by E_T. Secondly, we focus on compact sets $\mathcal{X} \subset \mathbb{R}^n$. A primary design choice in our method is to incorporate the set constraint as an additional indicator function to each prediction. This is equivalent to modeling each cost function as a composite function: $f_t(\boldsymbol{x}) + I_{\mathcal{X}}(\boldsymbol{x})$. The indicator part is then always assumed to be "predicted" perfectly. Nonetheless, the linearization of the past *composite* costs is now different. Namely, our proposed update becomes:

$$\boldsymbol{x}_{t+1} = \arg\min_{\boldsymbol{x}} \langle \boldsymbol{p}_{1:t}, \boldsymbol{x}\rangle + r_{1:t}(\boldsymbol{x}) + \tilde{f}_{t+1}(\boldsymbol{x}) + I_{\mathcal{X}}(\boldsymbol{x}), \tag{4.4}$$

with the state vector $\boldsymbol{p}_{1:t}$ calculated as the aggregation of

$$\begin{aligned} &\boldsymbol{p}_t = \boldsymbol{g}_t + \boldsymbol{g}_t^I, \\ &\boldsymbol{g}_t \in \partial f_t(\boldsymbol{x}_t), \quad \boldsymbol{g}_t^I \in \partial I_{\mathcal{X}}(\boldsymbol{x}_t) = \mathcal{N}_{\mathcal{X}}(\boldsymbol{x}_t), \end{aligned}$$

where $\mathcal{N}_{\mathcal{X}}(\boldsymbol{x})$ is the normal cone at $\boldsymbol{x}$, and is defined as

$$\mathcal{N}_{\mathcal{X}}(\boldsymbol{x}) \doteq \{\boldsymbol{g} \,|\, \langle \boldsymbol{g}, \boldsymbol{y} - \boldsymbol{x}\rangle \leq 0, \forall \boldsymbol{y} \in \mathcal{X}\}.$$

The simple yet key observation is that with the extra flexibility provided by $\boldsymbol{g}_t^I$, the state of the algorithm need not be the simple aggregation $\boldsymbol{g}_{1:t}$. Rather, some of the summands can be attenuated or *pruned* by carefully selecting $\boldsymbol{g}_t^I$ from the cone $\mathcal{N}_{\mathcal{X}}(\boldsymbol{x}_t)$, This is possible since $\mathcal{N}_{\mathcal{X}}(\boldsymbol{x}_t)$ contains all (scaled) negative subgradients of the expression in (4.4) when $\boldsymbol{x}_t$ lies on the boundary of $\mathcal{X}$ $\boldsymbol{x}_t \in \mathbf{bd}(\mathcal{X})$ (from the optimality conditions for constrained problems, see e.g., [113, Thm. 3.67]). In words, when an iterate leaves the feasible set and is thus projected back, we can choose to prune the state that led to this situation and replace it with an *alternative* state $\boldsymbol{p}_{1:t}$, that induces the same iterate but is smaller in norm. This construction is crucial, as we later show that the norm of the state is the key bottleneck for FTRL when competing with time-varying comparators. Fig. 4.1 illustrates how different state constructions behave upon a switch in cost direction; note that the FTRL update, with a fixed $r(\cdot)$ can be expressed as the projection of $\nabla r^\star(-\boldsymbol{g}_{1:t})$, where $r^\star(\cdot)$ is the conjugate of $r(\cdot)$ (see def. in Appendix 4A.1.4).

Our main contribution is formalizing this intuition to provide a dynamic regret analysis of Optimistic FTRL, leading to a new variant, *Optimistic Follow the Pruned Leader* (`OptFPRL`). This variant achieves *zero* dynamic regret when predictions are

[2] A proximal regularizer $r_t(\boldsymbol{x})$ is minimized at $\boldsymbol{x}_t$.

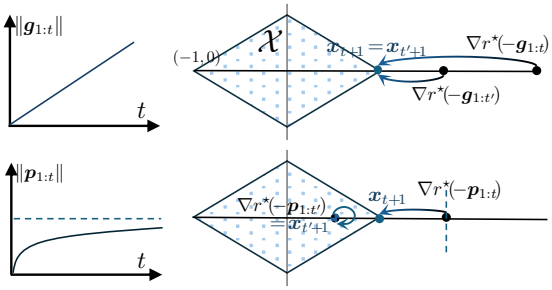


Figure 4.1: Effect of dual state size on iterates agility. We consider two slots $t < t'$, where gradients switch direction: $\boldsymbol{g}_\tau = (-1, 0)$ for $\tau \le t$, and $\boldsymbol{g}_\tau = (1, 0)$ for $\tau > t$. **Top**: Standard FTRL accumulates a large state $\boldsymbol{g}_{1:t}$, and the update via $\nabla r^\star(\boldsymbol{g}_{1:t})$ becomes insensitive to the change in direction; both $\boldsymbol{g}_{1:t}$ and $\boldsymbol{g}_{1:t'}$ map to the same iterate. **Bottom**: A well-maintained state $\boldsymbol{p}_{1:t}$ remains bounded, and hence its mapping stays close to $\mathcal{X}$, enabling $\boldsymbol{x}_{t'}$ to start aligning quickly with the new better iterate direction $(-1, 0)$.

4

perfect. This is because the quality of predictions controls *all* regret terms, including P_T. To our knowledge, this is the first variant to explicitly have such full dependence on prediction accuracy without oracle tuning. In the general case, `OptFPRL` also maintains the minimax optimal rate of $\mathcal{O}(\sqrt{(1+P_T)T})$ when P_T is known. We also present a version that does not require prior knowledge of P_T, but assumes observability, while still maintaining dependence on prediction errors.

Next, since `OptFPRL` follows an FTRL-style analysis, it is well-suited to incremental regularization in the manner of AdaFTRL [164]. In this scheme, we set the regularization *recursively*, incrementing it by, roughly, the regret by time t had we followed $\boldsymbol{x}_t, \forall \tau \le t$ (local regret). This is different from the more common approach, which sets this regularization as an *upper-bound* on this local regret. While the two are nearly equivalent in the worst case, the former allows for more refined bounds by applying only the minimal necessary regularization at each slot. Unlike prior optimistic dynamic regret algorithms, this style of regularization offers a more granular bound.

In summary, we investigate FTRL's dynamic regret over compact sets, and identify the unbounded growth of the state as the main bottleneck. By leveraging a simple and general mechanism, pruning past gradients, we gain a new degree of freedom that enables improved optimistic bounds.

4.3. Related Work

Optimism and (dynamic) regret. The pursuit of data dependence has been a central theme in online learning research since the introduction of AdaGrad [157, 165]. This has eventually led to the development of Optimistic FTRL (e.g., [26]). For a comprehensive survey of data-dependent online learning, we refer readers to [166]. However, these studies focus on static regret. Although the dynamic regret metric has been part of the OCO framework since its introduction, the first data-dependent dynamic regret bound only appeared in [160]. The authors established a general bound, from which the result $\mathcal{O}(\sqrt{(P_T^*+1)(E_T+1)}$ can be derived, where P_T^* is

specific to the sequence of minimizers comparators $\{\boldsymbol{u}_t = \arg\min_{\boldsymbol{x}} f_t(\boldsymbol{x})\}$, but in fact the bounds hold for any sequence whose path is *observable* online. This is done via a specialized doubling trick for non-monotone quantities. By removing the assumption of observable path lengths (i.e., considering all sequences simultaneously), dynamic regret bounds typically lose the sublinear dependence on P_T. For instance, in [160], the bound becomes $\mathcal{O}((P_T+1)\sqrt{E_T+1})$.

To address this, [167] proposed a meta-learning framework that achieves $\mathcal{O}(\sqrt{T(1+P_T)})$ for any sequence, which is shown to be minimax optimal. This framework was made data-dependent in [168], who obtained $\mathcal{O}(\sqrt{(V_T+P_T+1)(1+P_T)})$ bound, among others. In this framework, multiple sub-learners are employed, each implementing Online Gradient Descent (OGD) with a different (doubling) learning rate, tuned for various ranges of P_T. These sub-learners are then "tracked" by a variant of the Hedge meta-algorithm. This two-layer approach was later unified within the framework of Optimistic OMD and made more efficient in terms of the number of gradient queries [169]. Overall, knowing P_T in advance, or assuming it is observable online, allows us to tune the regularization (or learning rate for OMD) with this knowledge, getting the better dependence $\sqrt{P_T}$. Without such assumptions, we can use the meta-learning framework, which learns online the best such regularization/learning rate under smoothness assumptions.

Setting aside the order of dependence on P_T for a moment, we examine the quantity E_T. The aforementioned studies suffer the shortcoming hinted at in the introduction: to make E_T small, we require knowledge of $\nabla f_t(\boldsymbol{x}_t)$ at *the start* of time slot t, which is generally not feasible[3] even if access to $\nabla f_t(\cdot)$ was provided (i.e., perfect prediction). This is because $\boldsymbol{x}_t$ remains unknown at the start of t — it is the very point being determined. While the bounds are still optimistic, they are not informative for the predictor design. In the literature, this "cyclic" issue was only identified in [161, Sec. 1.B]. Thus, the authors introduce another related quantity D_T, defined earlier, and obtain $\mathcal{O}((1+P_T)(1+\sqrt{D_T}))$.[4] In the FTRL variants we propose, we do not have the cyclic issue because we do not require linearizing the predictions $\tilde{f}_t(\cdot)$.

The aforementioned studies are based on OMD, particularly on the "two-steps" variant of [158], where the learner selects two points in each iteration, an intermediate one and the actual action. This distinction from our work is not merely technical. For instance, in the related "OCO with delay" setting, prior work has established guarantees for Optimistic FTRL and the one-step version of Optimistic OMD proposed in [166, Sec. 7.2]. However, similar guarantees are not yet known for the two-step variant of OMD [86], and it remains unclear whether the same analysis can be extended to them.

Additionally, focusing again on optimism, a common limitation in the aforementioned studies is that the P_T quantity[5] in the bounds is unaffected by the predictions. That is, the P_T term may appear independently of any controllable quantity such

[3] Unless all functions are originally linear. Note that linearization does not solve this issue, as it is performed *after* the learner has committed to its action.

[4] The authors also derive bounds based on "temporal variation" and path length with approximate dynamical models, but we do not consider these quantities here.

[5] Or $\sqrt{P_T}$, assuming a known budget or an observable sequence with a doubling trick.

as E_T, leading to bounds of $\mathcal{O}(P_T)$ even under perfect predictions (or zero gradient variation). While this may partly reflect limitations in the existing analyses rather than the algorithms themselves, the current bounds do not fully capture the interaction between prediction quality and path length. In contrast, the FTRL variant we present shows that P_T, or more precisely, each $\|\boldsymbol{u}_{t+1} - \boldsymbol{u}_t\|$, is multiplied by the prediction error (e.g., ϵ_t or $\sqrt{E_t}$). Thus, the effect of P_T can be attenuated when predictions are accurate.

OMD, FTRL, and (linearized) history. The interplay between OMD and FTRL has received increasing attention in recent years. [170] studied a modified version of OMD under the static regret metric and showed its equivalence to Dual Averaging, a time-adaptive instance of FTRL, up to some terms in the normal cone. In this paper, we show that it is precisely these normal cone terms that become critical to achieving dynamic regret guarantees. For dynamic regret, [162] provides a comprehensive study via "centered" OMD, which incorporates FTRL-like centering properties. Their focus is primarily on unbounded domains, where such centering is essential. While their work integrates FTRL features into OMD, we take the opposite approach: extending native FTRL results to dynamic settings. This approach reveals failure modes of FTRL in non-stationary environments and offers principled solutions. Specifically, we investigate how modulating FTRL's state, in bounded domains, leads to regret bounds that are fully modulated by prediction accuracy.

In fact, our update is mostly related to the following form

$$\begin{aligned}\boldsymbol{x}_{t+1} &= \arg\min_{\boldsymbol{x}} \langle \boldsymbol{g}_{1:t} + \boldsymbol{g}^{\psi}_{1:t-1}, \boldsymbol{x}\rangle + r(\boldsymbol{x}) + I_{\mathcal{X}}(\boldsymbol{x}),\\ \boldsymbol{g}^{\psi}_{t} &\in \partial I_{\mathcal{X}}(\boldsymbol{x}_{t+1}),\end{aligned}$$

which appears in [112] under the name "FTRL Greedy", and was studied under the *static* regret metric. Beyond explicitly modeling the sub-gradient selection from the cone, which allows controlling when and what to prune, we extend this formulation by incorporating: (*i*) function predictions, (*ii*) prediction-adaptive regularization, and (*iii*) recursive regularization inspired by AdaFTRL. These modifications require different analysis tools, especially under the dynamic regret metric.

FTRL variants that reduce dependence on history have recently been proposed by [153, 171] using *geometric discounting* of all past costs, which is specifically designed for the metric of "discounted" regret and its applications. Interestingly, however, [153] also observes that such manipulation of FTRL's state can endow it with certain (not necessarily optimal) dynamic regret guarantees. Though their method differs, the core insight aligns with ours: limiting FTRL's memory is essential for adapting to non-stationarity.

We also note the existence of other approaches for modeling "optimism" in OCO, other than seeking E_T dependence, such as the Stochastically Extended Adversary (SEA) model [130] that was studied in [163] for OMD with dynamic comparators. In addition, there exists the "correlated hints" interpretation of optimism [172], where the prediction quality is measured by their correlation with the actual cost. This later line of work assumes *strongly* convex domains and obtains a $\mathcal{O}((1 +$

$P_T)\log^2(T)\sqrt{B})$ bound, where B is the number of slots without correlation. Lastly, an interesting reduction from dynamic to static settings is explored in [173], which revealed a fundamental tradeoff between gradient variability and the comparator sequence's complexity measures (e.g., the path length considered here).

4.4. OptFPRL

In this section, we present the proposed algorithm and characterize its dynamic regret. The routine of `OptFPRL` is described in Alg. 10. It takes as input the compact set $\mathcal{X}$, along with a strategy for determining the regularization parameters σ_t based on information available up to time t.

The initial action is based on the first prediction. Then, upon executing each $\boldsymbol{x}_t$ (line 3), the true cost $f_t(\boldsymbol{x})$ is revealed and the subgradient $\boldsymbol{g}_t$ is computable (line 5). In lines 6 and 7, we evaluate the prediction error ϵ_t, which is used to update the regularization parameter σ_t in the regularizer $r_t(\cdot)$ according to some pre-determined strategy.

We use scaled Euclidean regularizers of the form

$$r_t(\boldsymbol{x}) \doteq \frac{\sigma_t}{2}\|\boldsymbol{x}\|^2, \forall t \geq 1.$$

The regularizers are set such that $r_{1:t}$ is 1-strongly convex w.r.t. the scaled Euclidean norm $\|\cdot\|_t \doteq \sqrt{\sigma_{1:t}}\|\cdot\|$ whose dual norm is $\|\cdot\|_{t,*} = {}^1\!/\!_{\sqrt{\sigma_{1:t}}}\|\cdot\|$, hence we refer to σ_t also as the "strong" convexity parameters.

Next, we select $\boldsymbol{g}_t^I$ according to the following: for $t = 1$, set $\boldsymbol{g}_1^I = -\boldsymbol{g}_1$ if $\epsilon_1 = 0$, and $\boldsymbol{g}_1^I = 0$ otherwise. For all $t \geq 2$:

$$\boldsymbol{g}_t^I = \begin{cases} -(\boldsymbol{p}_{1:t-1} + \tilde{\boldsymbol{g}}_t + \sigma_{1:t-1}\boldsymbol{x}_t) & \text{if } \boldsymbol{x}_t^{\text{uc}} \notin \mathcal{X} \\ 0 & \text{otherwise,} \end{cases} \tag{4.5}$$

where $\boldsymbol{x}_t^{\text{uc}}$ is the unconstrained iterate obtained by solving (4.4) in $\mathbb{R}^n$ (i.e., without the indicator function).

To see why we can always set $\boldsymbol{g}_t^I$ as such, note that when $\boldsymbol{x}_t^{\text{uc}} \notin \mathcal{X}$, then $\boldsymbol{x}_t \in \mathbf{bd}(\mathcal{X})$ (the projection of a point outside a compact convex set lies on the boundary). Hence, $\mathcal{N}_{\mathcal{X}}(\boldsymbol{x}_t)$ contains vectors other than 0. In particular, it contains $-(\boldsymbol{p}_{1:t-1} + \tilde{\boldsymbol{g}}_t + \sigma_{1:t-1}\boldsymbol{x}_t)$ and in fact all positive multiples of this vector. This follows directly from the optimality condition for the constrained iterate in (4.4), (see, e.g., [113, Thm. 3.67]), and the definition of the normal cone. For the second case, 0 is always a valid subgradient of the indicator function since, in this case, $\boldsymbol{x}_t^{\text{uc}} = \boldsymbol{x}_t \in \mathcal{X}$.

Intuitively, this choice of linearization ensures an alternative state $\boldsymbol{p}_{1:t}$ (instead of $\boldsymbol{g}_{1:t}$). The former shall stop growing in norm compared to the latter beyond a certain t, since the action will hit the boundary of $\mathcal{X}$, starting the pruning thereafter. This will be formalized in the analysis, where we show that $\|\boldsymbol{p}_{1:t}\|$ cannot grow faster than the regularization, which is added by us optimistically (i.e., $\propto \sqrt{E_t}$). This is not true in general for an arbitrary choice of linearization, particularly for the default choice of using only $\boldsymbol{g}_t$.

We note that $\boldsymbol{g}_t^I$ need not be non-zero at every time slot t where $\boldsymbol{x}_t^{\text{uc}} \notin \mathcal{X}$. Instead, pruning of the accumulated state can be delayed for a fixed number of

Algorithm 10: Optimistic Follow the Pruned Leader(`OptFPRL`)

Input: Compact set $\mathcal{X}$, strategy for selecting $\sigma_t, \forall t$.
Output: $\{\boldsymbol{x}_t\}_{t=1}^T$.
1 set $\boldsymbol{x}_1 = \arg\min_x \tilde{f}_1(\boldsymbol{x}) + I_{\mathcal{X}}(x)$
for $t = 1, 2, \ldots, T$ **do**
2 Use action $\boldsymbol{x}_t$
3 *($f_t(\cdot)$ is revealed)*
4 Incur cost $f_t(\boldsymbol{x}_t)$ and compute $\boldsymbol{g}_t \in \partial f_t(\boldsymbol{x}_t)$
5 Compute $\tilde{\boldsymbol{g}}_t \in \partial \tilde{f}_t(\boldsymbol{x}_t)$ and the error $\epsilon_t = \|\boldsymbol{g}_t - \tilde{\boldsymbol{g}}_t\|$
6 Calculate the parameter σ_t using ϵ_t.
7 Set $\boldsymbol{g}_t^I$ according to (4.5).
8 Compute the (pruned) vector $\boldsymbol{p}_t = \boldsymbol{g}_t + \boldsymbol{g}_t^I$.
9 Receive prediction $\tilde{f}_{t+1}(\cdot)$.
10 Compute $\boldsymbol{x}_{t+1}^{\text{uc}}$ by solving (4.4) in $\mathbb{R}^n$.
11 Set $\boldsymbol{x}_{t+1} = \Pi(\boldsymbol{x}_{t+1}^{\text{uc}})$.
end

steps k, resulting in a *hybrid* state of the form $\boldsymbol{p}_{1:k-1} + \boldsymbol{g}_{k:t}$. This is discussed further in Appendix 4A.1.4.

In line (10), a prediction for the next cost is received, and the next unconstrained iterate is updated. Lastly, a feasible point is then recovered via a Euclidean[6] projection into $\mathcal{X}$.

Next, we characterize the dynamic regret of `OptFPRL` under different regularization strategies.

4.4.1. Dynamic Regret of OptFPRL

In this section, we explore different strategies for setting the regularization parameters σ_t, and analyze the resulting dynamic regret bounds. We begin by outlining the general setting shared across all regularization strategies.

Settings 1. Let $\mathcal{X} \subset \mathbb{R}^d$ be a compact, convex set such that $\|\boldsymbol{x}\| \leq R$ for all $\boldsymbol{x} \in \mathcal{X}$. Let $\{f_t(\cdot), \tilde{f}_t(\cdot)\}_{t=1}^T$ be any sequence of L-Lipschitz convex functions. Define the path length P_T as in (4.1), the cumulative prediction error E_T as in (4.2), and the hybrid term $H_T \doteq \sum_{t=1}^T \epsilon_t \|\boldsymbol{u}_{t+1} - \boldsymbol{u}_t\|$.

P_T-agnostic Regularization

The first regularization strategy we consider is the standard optimistic one, which sets σ_t such that $\sigma_{1:t} \propto \sqrt{E_T}$:

$$\begin{aligned} \sigma &= \frac{1}{4R}, \quad \sigma_1 = \sigma\epsilon_1, \\ \sigma_t &= \sigma\left(\sqrt{E_t} - \sqrt{E_{t-1}}\right), \forall t \geq 2. \end{aligned} \tag{4.6}$$

[6] Technically, the projection is w.r.t. $\|\cdot\|_t$. Since $\|\cdot\|_t$ is a scaled Euclidean, the result is the same. This is not true when $\|\cdot\|_t$ is induced by a general positive semidefinite matrix.

Theorem 4.1. *Under Settings 1, Alg. 10 run with the regularization strategy in* (4.6) *produces points* $\{\boldsymbol{x}_t\}_{t=1}^T$ *such that, for any* T*, the dynamic regret* $\mathcal{R}_T$ *satisfies:*

$$\begin{aligned}\mathcal{R}_T &\leq \big(5.8R + (1/2)P_T\big)\sqrt{E_T} + H_T \\ &= \mathcal{O}\left((1+P_T)\sqrt{E_T}\right).\end{aligned} \tag{4.7}$$

Remarks. We observe from (4.7) that all terms in the regret bound are modulated by the prediction errors. When the predictions are perfect, the bound reduces to zero (regardless of P_T), and it gracefully degrades as prediction errors grow. Note that by Cauchy-Schwarz and the boundness of $\mathcal{X}$, the hybrid term can be bounded as $H_T \leq \sqrt{2R}\sqrt{E_T P_T}$. Therefore, the overall bound is never worse than $\mathcal{O}((1+P_T)\sqrt{E_T})$, matching known OMD results when P_T is not accounted for, and tighter when predictions are accurate. In the static comparator case ($P_T = 0$), the bound recovers the standard $\mathcal{O}(\sqrt{E_T})$ result.

Regularization with Prior P_T Knowledge

In many cases, P_T can be provided to the algorithm a priori as a measure of the comparator's complexity (e.g., [154, 174]). That is, we wish to compete against any sequence whose path length is at most the provided value of P_T. With this given target, we can adjust the regularization to account for the expected nonstationarity and obtain better bounds, as outlined next:

$$\begin{aligned}&\sigma = \frac{1}{2\sqrt{2RP_T'}}, \quad P_T' \doteq 2R + P_T, \quad \sigma_1 = \sigma\epsilon_1, \\ &\sigma_t = \sigma\left(\sqrt{E_t} - \sqrt{E_{t-1}}\right), \forall t \geq 2.\end{aligned} \tag{4.8}$$

Theorem 4.2. *Under Settings 1, Alg. 10 run with the regularization strategy in* (4.8) *produces points* $\{\boldsymbol{x}_t\}_{t=1}^T$ *such that, for any* T*, the dynamic regret* $\mathcal{R}_T$ *satisfies:*

$$\begin{aligned}\mathcal{R}_T &\leq \left(4\sqrt{2R^2+P_T} + \frac{R}{8} + \sqrt{\frac{RP_T}{2}}\right)\sqrt{E_T} + H_T \\ &= \mathcal{O}\left((1+\sqrt{P_T})\sqrt{E_T}\right).\end{aligned}$$

Remarks. This regularization strategy preserves the full modulation by prediction errors while also matching the minimax-optimal bound $R_T = \Omega(\sqrt{(1+P_T)T})$ [167, Thm. 2] even when all predictions fail. This type of bound is new for FTRL-style algorithms. Furthermore, in its full dependency on E_T (i.e., without P_T or constant terms that are independent of E_T), it represents a refinement even compared to OMD. More broadly, access to a prior bound on P_T enables tailoring the regularization to the expected nonstationarity, allowing us to safeguard the minimax rate while still adapting to prediction accuracy.

Regularization with Unknown P_T

If P_T is unknown but observable, it can be estimated online alongside E_t. However, since $\sqrt{E_t/P_t}$ is no longer necessarily monotonic, we should safeguard against negative regularization coefficients. First, define the augmented seen path length at t as:

$$P'_t \doteq 2R + P_t = 2R + \sum_{\tau=1}^{t-1} \|\boldsymbol{u}_{\tau+1} - \boldsymbol{u}_\tau\|.$$

We adopt a regularization strategy that attempts to track $\sqrt{E_T/P_T}$ by its online estimate $\sqrt{E_t/P_t}$, while using a $\max(\cdot,\cdot)$ operator to ensure non-negative regularization.

$$\begin{aligned} \sigma &= \frac{1}{2\sqrt{2R}}, \quad \sigma_1 = \frac{\sigma\epsilon_1}{\sqrt{P'_1}}, \\ \sigma_t &= \sigma \max\left(0, \sqrt{\frac{E_t}{P'_t}} - \sqrt{\frac{E_{t-1}}{P'_{t-1}}}\right), \forall t \geq 2. \end{aligned} \tag{4.9}$$

Theorem 4.3. *Under Settings 1, Alg. 10 run with the regularization strategy in* (4.9) *produces points* $\{\boldsymbol{x}_t\}_{t=1}^T$ *such that, for any* T *the dynamic regret* $\mathcal{R}_T$ *satisfies:*

$$\begin{aligned} \mathcal{R}_T &\leq 5.5\sqrt{R}\sqrt{E_T P'_T} + H_T + \sqrt{R/2}\ A_T \\ &= \mathcal{O}\left((1+\sqrt{P_T})\sqrt{E_T} + A_T\right), \textit{where} \\ A_T &\doteq \sum_{t=1}^{T} \sum_{\tau \in [t]^+} \left(\sqrt{\frac{E_{\tau-1}}{P'_{t-1}}} - \sqrt{\frac{E_\tau}{P'_\tau}}\right) \|\boldsymbol{u}_{t+1} - \boldsymbol{u}_t\|, \\ [t]^+ &= \left\{ 2 \leq \tau \leq t \;\middle|\; \sqrt{\frac{E_{\tau-1}}{P'_{\tau-1}}} - \sqrt{\frac{E_\tau}{P'_\tau}} \geq 0 \right\}. \end{aligned}$$

Remarks. Note that this bound resembles that of Theorem 4.2, except for the additional term A_T, which arises due to the potential non-monotonicity of the estimated quantity $\sqrt{E_t/P_t}$. If this sequence were monotonic and non-decreasing, A_T would vanish since $[t]^+$ would be empty. This case allows us to recover the bound of Theorem 4.2 without requiring prior knowledge of P_T. In contrast, in the worst-case scenario, where the sequence alternates direction at every round, we obtain $A_T = \mathcal{O}(\sqrt{E_T}(P_T+1))$ (see Appendix 4A.3.3), leading to the looser bound in Theorem 4.1. A comparable correction term to A_T also appears in the optimistic OMD framework [161, Remark 2.19], and likewise depends on monotonicity, reaching $\sqrt{E_T}P_T$ in the worst case. The above-mentioned bound is, however, more interpretable. A standard workaround to this monotonicity issue is a doubling-trick variant [160], which, however, introduces slight problem-independence through a multiplicative $\mathcal{O}(R\log T)P_T$ factor that persists even under perfect predictions.

Recursive Regularization

In this subsection, we employ the regularization strategy of AdaFTRL [164], which sets the strong convexity parameters recursively, ensuring the minimal required regularization. Namely, we do not set σ_t such that $\sigma_{1:t} \propto \sqrt{E_t}$, as in the strategies discussed earlier, and prior works on optimistic dynamic regret. Instead, we set $\sigma_t \propto \delta_t \doteq h_{0:t-1}(\boldsymbol{x_t}) + \langle \boldsymbol{p}_t, \boldsymbol{x}_t \rangle - \min_{\boldsymbol{x}} \left(h_{0:t-1}(\boldsymbol{x}) + \langle \boldsymbol{p}_t, \boldsymbol{x} \rangle \right)$, where $h_t(\cdot)$ is the regularized loss: $h_t(\cdot) = \langle \boldsymbol{p}_t, \cdot \rangle + r_t(\cdot)$. This choice of σ_t is such that $\sigma_{1:t} \propto c \leq \sqrt{E_t}$. In other words, we increase the strong convexity exactly in proportion to the (regularized) cumulative loss observed at t (denoted as δ_t), rather than an upper bound on that loss.

Since the added strong convexity essentially determines the regret bound, this leads to tighter bounds overall that are no worse than the ones derived earlier. Namely, we use the following regularization strategy:

$$\sigma = \frac{1}{8R^2}, \quad \sigma_t = \sigma \delta_t, \quad \delta_1 = \langle \boldsymbol{g}_1, \boldsymbol{x}_1 \rangle - \min_{\boldsymbol{x} \in \mathcal{X}} \langle \boldsymbol{g}_1, \boldsymbol{x} \rangle$$
$$\delta_t = h_{0:t-1}(\boldsymbol{x_t}) + \langle \boldsymbol{p}_t, \boldsymbol{x}_t \rangle - \min_{\boldsymbol{x} \in \mathcal{X}} \left(h_{0:t-1}(\boldsymbol{x}) + \langle \boldsymbol{p}_t, \boldsymbol{x} \rangle \right), \quad \forall t \geq 2. \tag{4.10}$$

Theorem 4.4. *Under Settings 1, Alg. 10 run with the regularization strategy in* (4.10) *produces points* $\{\boldsymbol{x}_t\}_{t=1}^T$ *such that, for any* T*, the dynamic regret* $\mathcal{R}_T$ *satisfies:*

$$\begin{aligned} \mathcal{R}_T &\leq 1.1\ \delta_{1:T} + \sum_{t=1}^{T-1} \frac{1}{4R} \delta_{1:t} \|\boldsymbol{u}_{t+1} - \boldsymbol{u}_t\| + H_T \\ &\leq (3.7R + P_T)\sqrt{E_T} + H_T = \mathcal{O}\left((1 + P_T)\sqrt{E_T} \right). \end{aligned}$$

Remarks. Since the δ_t terms are often smaller than their upper bound, this minimal regularization approach is particularly advantageous in dynamic environments, where regularization terms sum is nested within the primary sum over $[T]$. While this tuning strategy has been employed previously in *static* settings for optimistic learning [86], it has not yet been leveraged for optimism in dynamic settings[7]. Nonetheless, the recursive nature of this regularization adds a technical challenge since the accumulated strong convexity does not have a closed-form expression in terms of $\sqrt{E_t}$. Fortunately, this recursion is still shown to be bounded by $\mathcal{O}(\sqrt{E_t})$ via tools developed in the AdaFTRL framework.

4.5. Tools for FTRL's Dynamic Regret Analysis

In this section, we present the primary analytical tools used to derive the regret bounds of the previous section. These results extend traditional FTRL analyses to incorporate both optimism and dynamic comparators. First, we bound the regret

[7] We note, however, the related temporal-variation-based dynamic regret bound in Implicit OMD [175], where the structure of "function changes", rather than "prediction errors", is exploited.

via linearization:

$$\mathcal{R}_T = \sum_{t=1}^{T} j_t(\boldsymbol{x}_t) - j_t(\boldsymbol{u}_t) \leq \sum_{t=1}^{T} \langle \boldsymbol{p}_t, \boldsymbol{x}_t - \boldsymbol{u}_t \rangle,$$

where $j_t(\boldsymbol{x}) \doteq f_t(\boldsymbol{x}) + I_{\mathcal{X}}(\boldsymbol{x})$. The inequality follows directly from the convexity of $j_t(\cdot)$,[8] noting that $\boldsymbol{p}_t \in \partial j_t(\boldsymbol{x}_t)$ by definition of $\boldsymbol{p}_t$ (recall $\boldsymbol{p}_t = \boldsymbol{g}_t + \boldsymbol{g}_t^I$, with $\boldsymbol{g}_t \in \partial f_t(\boldsymbol{x}_t), \boldsymbol{g}_t^I \in \partial I_{\mathcal{X}}(\boldsymbol{x}_t)$).

Our analysis begins with a generalization of the main FTRL lemma [112, Lem. 5].

Lemma 4.5. *(Strong Dynamic Optimistic FTRL). Let $\{f_t(\cdot), \tilde{f}_t(\cdot), \boldsymbol{u}_t\}_{t=1}^T$ be an arbitrary set of functions, predicted functions, and comparators within $\mathcal{X}$, respectively. Let $r_t(\cdot)$ be non-negative regularization functions such that*

$$\boldsymbol{x}_{t+1} \doteq \arg\min_{\boldsymbol{x}} h_{0:t}(\boldsymbol{x}) + \tilde{f}_{t+1}(\boldsymbol{x})$$

is well-defined, where $h_0(\boldsymbol{x}) \doteq I_{\mathcal{X}}(\boldsymbol{x})$, and $\forall t \geq 1$:

$$h_t(\boldsymbol{x}) \doteq \langle \boldsymbol{p}_t, \boldsymbol{x} \rangle + r_t(\boldsymbol{x}), \quad \boldsymbol{p}_t \in \partial j_t(\boldsymbol{x}_t).$$

Then, the algorithm that selects the actions $\boldsymbol{x}_{t+1}, \forall t$ achieves the following dynamic regret bound:

$$\mathcal{R}_T \leq \sum_{t=1}^{T} \overbrace{h_{0:t}(\boldsymbol{x}_t) - h_{0:t}(\boldsymbol{x}_{t+1}) - r_t(\boldsymbol{x}_t)}^{(I)} + \sum_{t=1}^{T-1} \underbrace{h_{0:t}(\boldsymbol{u}_{t+1}) - h_{0:t}(\boldsymbol{u}_t)}_{(II)} + r_t(\boldsymbol{u}_t).$$

The regret is thus decomposed into *three* main parts. Part **(I)** measures the penalty incurred due to not knowing $\boldsymbol{g}_t$ when deciding $\boldsymbol{x}_t$. The second part **(II)** measures the penalty incurred due to the non-stationarity of the environment (change of comparators). The last part, $r_t(\boldsymbol{u}_t)$, is a user-controlled quantity that reflects the amount of regularization introduced and will be traded off against the other terms in the bound that benefit from more regularization.

Next, we describe the upper bounds:

$$\textbf{(I)} \leq \min\left(\frac{1}{2}\|\boldsymbol{g}_t - \tilde{\boldsymbol{g}}_t\|_{t-1,*}^2, 2R\epsilon_t\right)$$
$$\textbf{(II)} \leq \left(\|\boldsymbol{p}_{1:t}\| + R\sigma_{1:t}\right)\|\boldsymbol{u}_{t+1} - \boldsymbol{u}_t\|$$

(I) is a result of a generalization of a common tool in OCO which bounds the difference in the value of a strongly convex function ($h_{0:t}(\cdot)$) when evaluated at a "partial"[9] minimizer ($\boldsymbol{x}_t$) versus the global minimizer (say, $\boldsymbol{y}$, and hence $\boldsymbol{x}_{t+1}$ also), which we state below.

[8] see, e.g., [156, Sec. 2.4], for more details on linearization.
[9] Recall that $\boldsymbol{x}_t$ does not minimize $h_{0:t}$, but a related function.

Lemma 4.6. *Let each function $h_{0:t}(\cdot)$ be 1-strongly convex w.r.t. a norm $\|\cdot\|_t$ defined as in Lemma 4.5. Let $\boldsymbol{x}_t = \arg\min_{\boldsymbol{x}} h_{0:t-1}(\boldsymbol{x}) + \tilde{f}_t(\boldsymbol{x})$. Then, we have the inequality*

$$h_{0:t}(\boldsymbol{x}_t) - h_{0:t}(\boldsymbol{x}_{t+1}) - r_t(\boldsymbol{x}_t) \leq \frac{1}{2}\|\boldsymbol{g}_t - \tilde{\boldsymbol{g}}_t\|^2_{t-1,*} \,.$$

The second term of the min is in fact a crude bound on the per-slot loss of $\boldsymbol{x}_t$ compared to the omniscient $\boldsymbol{x}_{t+1}$. We detail both bounds in Appendix 4A.2.

(II) follows directly from the first-order inequality for convex functions applied to $h_{0:t}(\cdot)$. Note that it is exactly the $\|\boldsymbol{p}_{1:t}\|$ term that explains the potential failure of FTRL in dynamic environments, even with a *constant* path length. Specifically, a trivial bound on this norm is linear in t and hence is super-linear in T even with one switch in the comparators. This is precisely the "vulnerability" that is exploited in the impossibility result in [162, Thm. 2]. Next, we show how pruning can *tie* this $\|\boldsymbol{p}_{1:t}\|$ term to the regularization parameters $\sigma_{1:t}$ (which we control), thus ensuring its sub-linearity.

Lemma 4.7. *(Optimistically Bounded State) Let $\{\boldsymbol{p}_t\}_{t=1}^T$ be a sequence of vectors such that each $\boldsymbol{p}_t = \boldsymbol{g}_t + \boldsymbol{g}_t^I$, where $\boldsymbol{g}_t \in \partial f_t(\boldsymbol{x}_t)$, and $\boldsymbol{g}t^I \in \partial I_{\mathcal{X}}(\boldsymbol{x}_t)$ is chosen according to the construction in* (4.5)*. That is, $\boldsymbol{p}_t$ corresponds to the linearization of the composite function $f_t(\cdot) + I_{\mathcal{X}}(\cdot)$ around $\boldsymbol{x}_t$. Then, for any t, the following holds:*

$$\|\boldsymbol{p}_{1:t}\| \leq R\sigma_{1:t-1} + \epsilon_t.$$

Proof. First, we begin by showing that

$$\text{For any } t,\ \boldsymbol{x}_t^{\text{uc}} \in \mathcal{X} \implies \|\boldsymbol{p}_{1:t}\| \leq \sigma_{1:t-1}R + \epsilon_t. \tag{4.11}$$

Since $\boldsymbol{x}_t^{\text{uc}} \in \mathcal{X}$, we know that $\boldsymbol{x}_t = \Pi(\boldsymbol{x}_t^{\text{uc}}) = \boldsymbol{x}_t^{\text{uc}}$ (The projection of a point within the set is itself), and hence $\boldsymbol{x}_t$ is a minimizer of the unconstrained update rule too:

$$\boldsymbol{x}_t = \arg\min_{\boldsymbol{x}\in\mathbb{R}^n} \langle \boldsymbol{p}_{1:t-1}, \boldsymbol{x}\rangle + r_{1:t-1}(\boldsymbol{x}) + \tilde{f}_t(\boldsymbol{x}) \tag{4.12}$$

From the first-order optimality condition for unconstrained problems (e.g., [113, Thm. 3.63]), we know that 0 is an element of the subdifferential of (4.12) at $\boldsymbol{x}_t$. Thus, $\exists \tilde{\boldsymbol{g}}_t \in \partial f_t(\boldsymbol{x}_t)$ such that

$$\begin{aligned} 0 &= \boldsymbol{p}_{1:t-1} + \sigma_{1:t-1}\boldsymbol{x}_t + \tilde{\boldsymbol{g}}_t \\ &\implies \boldsymbol{p}_{1:t-1} = -\sigma_{1:t-1}\boldsymbol{x}_t - \tilde{\boldsymbol{g}}_t, \end{aligned} \tag{4.13}$$

and we have that the norm of the state $\boldsymbol{p}_{1:t}$ satisfies

$$\begin{aligned} \|\boldsymbol{p}_{1:t}\| &= \|\boldsymbol{p}_{1:t-1} + \boldsymbol{p}_t\| \\ &\overset{(a)}{=} \|\boldsymbol{p}_{1:t-1} + \boldsymbol{g}_t\| \overset{(b)}{=} \|-\sigma_{1:t-1}\boldsymbol{x}_t - \tilde{\boldsymbol{g}}_t + \boldsymbol{g}_t\| \\ &\leq \|-\sigma_{1:t-1}\boldsymbol{x}_t\| + \|\boldsymbol{g}_t - \tilde{\boldsymbol{g}}_t\| \end{aligned}$$

$$= \sigma_{1:t-1}\|\boldsymbol{x}_t\| + \epsilon_t \overset{(c)}{\leq} \sigma_{1:t-1}R + \epsilon_t,$$

where (a) holds because $\boldsymbol{g}_t^I = 0$ (from (4.5) & $\boldsymbol{x}_t^{\text{uc}} \in \mathcal{X}$), (b) from (4.13), and (c) from the set size.

Next, we show that when $\boldsymbol{x}_t^{\text{uc}} \notin \mathcal{X}$, the state is forced via pruning to be bounded:

$$\text{For any } t,\ \boldsymbol{x}_t^{\text{uc}} \notin \mathcal{X} \implies \|\boldsymbol{p}_{1:t}\| \leq \sigma_{1:t-1}R + \epsilon_t. \tag{4.14}$$

When $\boldsymbol{x}_t^{\text{uc}} \notin \mathcal{X}$, the pruning condition is activated, which ensures that

$$\begin{aligned}
\boldsymbol{p}_{1:t} &= \boldsymbol{p}_{1:t-1} + \overbrace{\boldsymbol{g}_t \underbrace{-\boldsymbol{p}_{1:t-1} - \tilde{\boldsymbol{g}}_t - \sigma_{1:t-1}\boldsymbol{x}_t}_{\boldsymbol{g}_t^I}}^{\boldsymbol{p}_t} \\
&= \boldsymbol{g}_t - \tilde{\boldsymbol{g}}_t - \sigma_{1:t-1}\boldsymbol{x}_t. \\
\text{Thus, } \|\boldsymbol{p}_{1:t}\| &= \|\boldsymbol{g}_t - \tilde{\boldsymbol{g}}_t - \sigma_{1:t-1}\boldsymbol{x}_t\| \\
&\leq \|\boldsymbol{g}_t - \tilde{\boldsymbol{g}}_t\| + \sigma_{1:t-1}\|\boldsymbol{x}_t\| \\
&\leq \epsilon_t + \sigma_{1:t-1}R.
\end{aligned}$$

The lemma statement follows by (4.11) and (4.14). □

4.5.1. Regret Bound Derivation

We show the proof of Theorem 4.1 since it illustrates how the presented tools come together to obtain the results. It also provides a sufficient basis for sketching the proofs of the remaining theorems.

Proof of Theorem 4.1. We begin from the main lemma, Lemma 4.5, with the bounds on **(I)** and **(II)** terms:

$$\begin{aligned}
\mathcal{R}_T &\leq \sum_{t=1}^{T}\left(\min\left(\frac{1}{2}\|\boldsymbol{g}_t - \tilde{\boldsymbol{g}}_t\|^2_{t-1,*}, 2R\epsilon_t\right) + r_t(\boldsymbol{u}_t)\right) \\
&\quad + \sum_{t=1}^{T-1}\left(\left(R\sigma_{1:t} + \|\boldsymbol{p}_{1:t}\|\right)\|\boldsymbol{u}_{t+1} - \boldsymbol{u}_t\|\right) \qquad (4.15) \\
&\overset{(a)}{\leq} \sum_{t=1}^{T}\min\left(\frac{1}{2}\|\boldsymbol{g}_t - \tilde{\boldsymbol{g}}_t\|^2_{t-1,*}, 2R\epsilon_t\right) + \frac{R^2}{2}\sigma_{1:T} \\
&\quad + \sum_{t=1}^{T-1}\left(\left(2R\sigma_{1:t} + \epsilon_t\right)\|\boldsymbol{u}_{t+1} - \boldsymbol{u}_t\|\right) \\
&= \sum_{t=1}^{T}\min\left(\frac{\epsilon_t^2}{2\sigma\sqrt{E_{t-1}}}, 2R\epsilon_t\right) + \frac{R^2}{2}\sigma\sqrt{E_T} \\
&\quad + \sum_{t=1}^{T-1}\left(\left(2R\sigma\sqrt{E_t} + \epsilon_t\right)\|\boldsymbol{u}_{t+1} - \boldsymbol{u}_t\|\right) \qquad (4.16)
\end{aligned}$$

$$\overset{(b)}{\leq} 4\sqrt{2}R\sqrt{E_T} + \frac{R}{8}\sqrt{E_T} + \sum_{t=1}^{T-1}\frac{1}{2}\sqrt{E_t}\|\boldsymbol{u}_{t+1} - \boldsymbol{u}_t\| + H_T$$

$$\overset{(c)}{\leq} 5.8R\sqrt{E_T} + \frac{\sqrt{E_T}}{2}P_T + H_T$$

where (a) follows from the boundedness of $\mathcal{X}$, the state bound in Lemma 4.7, and the fact that $\sigma_{1:t-1} \leq \sigma_{1:t}$; (b) follows from applying Lemma 4A.9 (bounding the sum of non-increasing terms) and substituting the expression for σ; (c) uses that $\sqrt{E_t}$ is non-decreasing. □

Overall, after using the results developed in Sec. 4, obtaining the exact results of the theorems mainly hinges on available tools from the OCO literature on bounding the sum of non-increasing functions.

Proof sketch of other Theorems. The proofs of the other theorems follow similar main steps, and are detailed in the Appendix. When the σ parameter is normalized by $\sqrt{P_T}$, it can be seen from (4.16) that the P_T term will also be divided by the same, recovering the result of Theorem 4.2.

For Theorem 4.3, we write the sum $\sigma_{1:t}$ in (4.15) as the non-monotone $\sqrt{E_t/P'_t} - \sqrt{E_{t-1}/P'_{t-1}}$ provided that we add the corrective term A_T defined earlier. The former tracks the desired quantity $\sqrt{E_T/P'_T}$, which is what is required (and was used in Theorem 4.2) to recover the $\sqrt{E_T P_T}$ dependence. The corrective sum A_T is handled independently.

Lastly, for Theorem 4.4, we start again from Lemma, 4.5, but instead of using the upper bound of the term **(I)**, we use the δ_t terms. This results in a recursion whose solution is fortunately available among AdaFTRL lemmas.

4.6. Conclusion

This paper introduced an optimistic FTRL variant with new data-dependent dynamic regret guarantees, extending classical FTRL results and advancing our understanding of this foundational framework in non-stationary environments. These bounds are explicitly tied to the accuracy of predictions, offering refined performance guarantees in dynamic settings. The gist of our proposal lies in a simple pruning rule that modulates the memory (or state) of FTRL based on the quality of the obtained decisions, preventing the accumulation of redundant (negative) gradients that align with iterates on the boundary of the decision set. This technique can be extended to more flexible pruning strategies that control pruning magnitude or introduce additional pruning conditions, enabling a spectrum of algorithms ranging from fully "lazy" to fully "agile" iterates. As in the previous chapters, the environment in this chapter is modeled such that the feedback at each time slot depends solely on the current action, without regard to past actions. The following chapters relax this assumption, considering more general environments where the cost at each slot depends on the history of actions. Accordingly, a different performance metric will be employed to capture this added complexity.

5

Optimism In Presence of Memory

In this chapter, we shift our focus to optimistic online learning *with memory*. Specifically, we modify the feedback model in the online learning framework so that the cost at each time step depends not only on the current decision, but also on past decisions. Our primary focus is on the recently introduced *Non-Stochastic Control* (NSC) paradigm, a specific instance of online learning with memory. A drawback (and improvement opportunity) in NSC is the lack of algorithms that possess (*i*) adaptivity to environmental adversity (i.e., data-dependence) and (*ii*) adaptivity to predictions (i.e., optimism). The former refers to the inability of existing algorithms to adjust their strategies based on observed cost sequences (e.g., their variability), leading to bounds that rely on loose worst-case assumptions rather than the actual structure of encountered costs. The latter reflects the fact that NSC has yet to explore the integration of untrusted predictions to accelerate learning. Importantly, establishing data dependence (i.e., addressing the first limitation) is often a prerequisite for designing optimistic algorithms (i.e., addressing the second limitation).

To address these gaps, this chapter is structured into two main parts. First, we develop NSC algorithms that are adaptive to environmental adversity, ensuring that their operation is data-dependent and scales with the actual cost sequences rather than worst-case estimates. In the second part, we introduce the first optimistic algorithm for the NSC problem, leveraging untrusted predictions to enhance performance while maintaining robustness in adversarial settings. By bridging these two

The content of this chapter has been published in

- Naram Mhaisen and George Iosifidis. "Adaptive Online Non-stochastic Control". In *Proceedings of Learning for Dynamics and Control (L4DC)*. 2024.
- Naram Mhaisen and George Iosifidis. "Optimistic Online Non-stochastic Control via FTRL". In *Proceedings of IEEE Conference on Decision and Control (CDC)*. 2024.

missing components, this chapter advances the theoretical understanding of online learning with memory and extends NSC beyond its initial formulation.

5.1. Introduction & NSC Background

This chapter tackles the Online Non-stochastic Control problem: find a *policy* that endures minimum cost while controlling a *dynamical system* whose *state* changes via an unknown combination of learner's actions and external parameters. Optimal Non-stochastic control has significant applications ranging from the control of medical equipment [176] to energy management in data centers [177]. This work advances the results on this fundamental problem by proposing optimal control algorithms based on adaptive online learning.

Formally, we consider a typical NSC problem with a time-slotted dynamical system [178]. Namely, at each time step, the controller observes the system state $\boldsymbol{x}_t \in \mathbb{R}^{d_x}$ and decides an action $\boldsymbol{u}_t \in \mathbb{R}^{d_u}$ which induces cost $c_t(\boldsymbol{x}_t, \boldsymbol{u}_t)$. Then, the system transitions to state $\boldsymbol{x}_{t+1}$. Note that the new state and cost function, at each step, are revealed to the controller *after* it commits its action. Similar to [178], we study Linear Time Invariant (LTI) systems where the transition is parametrized by matrices A, B and a *disturbance* vector $\boldsymbol{w}_t$:

$$\boldsymbol{x}_{t+1} = A\boldsymbol{x}_t + B\boldsymbol{u}_t + \boldsymbol{w}_t. \tag{5.1}$$

We allow $\boldsymbol{w}_t$ to be arbitrarily set by an *adversary* that aims to manipulate the state transition, and we only restrict it to be universally upper-bounded, i.e., $\|\boldsymbol{w}\| \leq w$. Similarly, the adversary is allowed to select at each step *any* Lipschitz continuous convex cost function $c_t : \mathbb{R}^{d_x} \times \mathbb{R}^{d_u} \mapsto \mathbb{R}$.

The controller's task is to deduce a (possibly non-stationary) policy that maps states to actions, $\pi : \boldsymbol{x} \mapsto \boldsymbol{u}$, from a policy set denoted Π, leading to a trajectory of low costs $\{c_t(\boldsymbol{x}_t, \boldsymbol{u}_t)\}_{t=1}^T$. The employed performance metric in this setting is the *policy regret* [85], which measures the accumulated extra cost endured by the learner's policy compared to a stationary cost-minimizing policy designed with access to all future cost functions and disturbances:

$$\mathcal{R}_T \doteq \sum_{t=1}^{T} c_t\left(\boldsymbol{x}_t, \boldsymbol{u}_t\right) - \min_{\pi \in \Pi} \sum_{t=1}^{T} c_t\big(\boldsymbol{x}_t(\pi), \boldsymbol{u}_t(\pi)\big) \tag{5.2}$$

where $\boldsymbol{x}_t(\pi), \boldsymbol{u}_t(\pi)$ are the *counterfactual* state-action sequences that would have been generated under the benchmark policy, whereas $(\boldsymbol{x}_t, \boldsymbol{u}_t)$ are the *actual* state-action pair that resulted from following possibly different series of policies. A sublinear regret $\mathcal{R}_T = o(T)$ guarantees that the cost endured by the learner will converge to that of the optimal policy, i.e., $\mathcal{R}_T/T \to 0$, and the recent Gradient Perturbation Controller (GPC), proposed in [178], attains indeed $\mathcal{R}_T = \mathcal{O}(\sqrt{T})$. GPC's performance is established via a reduction to the Online Convex Optimization (OCO) with Memory framework [179], which in turn established sublinear regret via a reduction to the standard OCO framework [180].

Chapter notation. We denote scalars by small letters, vectors by bold small letters, and matrices by capital letters. Time indexing is done via a subscript. We

denote by $\{\boldsymbol{a}_t\}_{t=1}^T$ the set $\{\boldsymbol{a}_1, \ldots, \boldsymbol{a}_T\}$. $M = [M^{[i]}|M^{[j]}]$ denotes the augmentation of the matrices $M^{[i]}$ and $M^{[j]}$. $\|\cdot\|$ denotes the ℓ_2 norm for vectors and the Frobenius norm for matrices. $\|\cdot\|_*$ is the dual norm.$\langle\cdot,\cdot\rangle$ is the dot product for vectors and the Frobenius product for matrices. $\|\cdot\|_{\text{op}}$ is the matrix spectral norm (the induced ℓ_2 norm). We use $h_{a:b}$ to indicate $\sum_{s=a}^{b} h_s$ when s is irrelevant, and $f(\cdot)$ when the function's argument are irrelevant.

5.2. PRELIMINARIES

We introduce here the technical setup of NSC tailored to the assumptions and goals of this chapter.

Policy class. We consider the class Π of Disturbance Action Controllers (DAC) that was introduced in [178]. A policy $\pi \in \Pi$, with memory length p, is parametrized by p matrices $M \doteq \left[M^{[1]}|M^{[2]}|\ldots|M^{[j]}|\ldots|M^{[p]}\right]$, and a fixed stabilizing controller K. We also define the set $\mathcal{M} \doteq \{M : \|M\| \leq \kappa_M\}$, which uses the standard bounded variable assumption. The action at a step t according to a policy $\pi_t \in \Pi$, is then calculated via:

$$\boldsymbol{u}_t = K\boldsymbol{x}_t + \sum_{j=1}^{p} M_t^{[j]} \boldsymbol{w}_{t-j}. \tag{5.3}$$

Strong stability. This assumption is standard in OCO-based control as it enables non-asymptotic analysis [181, Def. 3.1]. It ensures the existence of a stabilizing controller K, such that $\|(A+BK)^t\|_{\text{op}} < \kappa(1-\delta)^t, \delta \in (0,1]$, $\kappa > 0$. Here, we assume that $\|A\|_{\text{op}} \leq 1-\delta$, which allows us to satisfy the stability assumption with K being the zero matrix. This simplification facilitates the analysis, but the obtained results are still extensible for nonzero K since K is an external parameter to NSC algorithms, see, e.g., the discussion in [182, Remark 4.1]. We also assume $\|B\| \leq \kappa_B$, while the boundedness of $\|A\|$ follows from its spectral norm bound.

Cost functions. We consider the family of general convex functions for the losses, and we denote with $G_t(M)$ the gradient matrix[1] of the cost $c_t(\boldsymbol{x}, \boldsymbol{u})$ w.r.t. M. If the argument M is not relevant (e.g., c_t is linear) or is fixed to M_t, then we denote the gradient simply with G_t. We also use the standard l-Lipschitz assumption $|c_t(\boldsymbol{x}, \boldsymbol{y}) - c_t(\boldsymbol{x}', \boldsymbol{y}')| \leq l\|(\boldsymbol{x}, \boldsymbol{u}) - (\boldsymbol{x}', \boldsymbol{u}')\|, \forall t \in [T]$.

DAC rationale. The DAC class strikes a balance between efficiency and performance. Specifically, both the states and actions are convex in the optimization variables M. This can be directly seen from (5.3) for the actions, whereas the state can be shown to be linear by unrolling the dynamics equation in (5.1) (e.g., see [178, Lem. 4.3]) which we adapt below to our notation:

Lemma 5.1. *Assuming that $\boldsymbol{x}_1 = 0$, and parameters $M_t, \boldsymbol{w}_t$ are 0 for $t \leq 0$, the state of the system reached at $t+1$ upon the execution of actions $\{\boldsymbol{u}_i\}_{i=1}^t$, derived from a DAC policy π_t, is:*

$$\boldsymbol{x}_{t+1} = \sum_{i=0}^{t} A^i \left(B \sum_{j=1}^{p} \left(M_{t-i}^{[j]} \; \boldsymbol{w}_{t-i-j} \right) + \boldsymbol{w}_{t-i} \right). \tag{5.4}$$

[1] Since our decision variable is in $\mathcal{M}$, the gradient "vector" can be organized into a matrix.

Clearly, $c_t(\boldsymbol{x}_t, \boldsymbol{u}_t)$ is convex in M since $\boldsymbol{x}_t$ and $\boldsymbol{u}_t$ are linear in variables M. This facilitates the minimization of the cost function. Note that the boundness on the norm of $\boldsymbol{x}$ and $\boldsymbol{u}$,[2] along with the Lipschitzness of $c(\cdot, \cdot)$ implies the existence of g s.t. $\|G_t\| \leq g, \forall t$.

Besides allowing convex costs, DAC policies are expressive as they approximate the class of linear policies up to an arbitrarily small constant error ζ . This follows from the next lemma from [82, Lem. 6.9].

Lemma 5.2. *Let $\|A\|_{op} = 1 - \delta, \|\boldsymbol{w}\| \leq w$. Then, for any linear policy $\pi^{\mathbb{L}}$ with $\|K\| \leq \kappa^{\mathbb{L}}$, and an arbitrarily small constant ζ, there is a DAC policy with $p = \lfloor 1/\delta \log\left(\kappa^{\mathbb{L}} w/\delta\zeta\right) \rfloor$ that achieves a cost at most $\mathcal{O}(\zeta)$ far from the cost of the linear policy:*

$$|c_t\big(\boldsymbol{x}_t(\pi), \boldsymbol{u}_t(\pi)\big) - c_t\big(\boldsymbol{x}_t(\pi^{\mathbb{L}}), \boldsymbol{u}_t(\pi^{\mathbb{L}})\big)| = \mathcal{O}(\zeta).$$

Approximating the large class of linear controllers through DAC is important because this class is guaranteed to include, for instance, the universally optimal controller in Linear Quadratic Regulator (LQR) settings, as well as linear positive systems with linear objective functions [183].

Policy regret. We proceed to provide more details on the regret of a policy, mentioned earlier in (5.2). We start with the benchmark DAC policy $\pi_\star \in \Pi$, that is fully characterized by matrix $M_\star$ which can be calculated by solving:

$$\begin{aligned}
\mathbf{P}_1 : \underset{M \in \mathcal{M}}{\text{minimize}} \quad & \sum_{t=1}^{T} c_t\big(\boldsymbol{x}_t(\pi_\star), \boldsymbol{u}_t(\pi_\star)\big), \\
\text{subject to} \quad & \boldsymbol{x}_{t+1} = A\boldsymbol{x}_t + B\boldsymbol{u}_t + \boldsymbol{w}_t, \qquad \forall t \in [T], \qquad (5.5) \\
& \boldsymbol{u}_t = K\boldsymbol{x}_t + \sum_{j=1}^{p} M^{[j]} \boldsymbol{w}_{t-j}, \quad \forall t \in [T]. \qquad (5.6)
\end{aligned}$$

Constraints (5.5) enforce the LTI dynamics of the state transition, and (5.6) ensure the actions are taken from a DAC policy. Clearly, this hypothetical policy can only be calculated with access to future disturbances and costs, whereas the learner, at each step t, has access only to information until $t-1$.

There are some important notation remarks in order here. Recall that π_t is the policy at step t, where the learner decides $\boldsymbol{u}_t$ using M_t. The state reached at t depends on the *sequence* of policies $\pi_1, \pi_2, \ldots, \pi_{t-1}$, also referred to as non-stationary policy, and we denote it with $\boldsymbol{x}_t\left(\pi_{1,\ldots,t-1}\right)$. In contrast, the *counterfactual* state reached at step t by following a stationary policy π at all steps up to t is denoted $\boldsymbol{x}_t(\pi)$.[3] Similarly, the action at t would in general differ for a stationary policy π versus a non-stationary policy, i.e., $\boldsymbol{u}_t(\pi_t) \neq \boldsymbol{u}_t(\pi_{1,\ldots,t})$. However, when K is the zero matrix, these vectors are equivalent. To reduce clutter when possible, we omit

[2]The bound on the state is due to the strong stability assumption and will be explicit in the analysis.

[3]We call $\boldsymbol{x}_t(\pi)$ counterfactual since it is not necessarily equal to the true state of the system $\boldsymbol{x}_t\left(\pi_{1,\ldots,t-1}\right)$.

the argument $\pi_{1,\dots,t-1}$ from $\boldsymbol{x_t}(\pi_{1,\dots,t-1})$. However, when we use the hypothetical state resulting from a fixed policy π, we make this explicit by writing $\boldsymbol{x}_t(\pi)$. The same applies to $\boldsymbol{u}_t$. Now, we define the policy regret as the cumulative difference between the cost induced by $\pi_\star$, and the cost of the non-stationary policy as:

$$\mathcal{R}_T(\pi_{1,\dots,T}, \pi_\star) \doteq \sum_{t=1}^{T} \Big(c_t\big(\boldsymbol{x}_t(\pi_{1,\dots,t-1}), \boldsymbol{u}_t(\pi_{1,\dots,t})\big) - c_t\big(\boldsymbol{x}_t(\pi_\star), \boldsymbol{u}_t(\pi_\star)\big)\Big).$$

Because of Lemma 5.2, $\pi_{1,\dots,T}$ has also a regret guarantee against the best policy in the linear class. That is, there is a constant $a > 0$, that depends only on l, $\kappa^{\mathbb{L}}$, and δ, such that $\mathcal{R}_T(\pi_{1,\dots,T}, \pi_\star^{\mathbb{L}}) = \mathcal{R}_T(\pi_{1,\dots,T}, \pi_\star) + a\zeta T$. Hence, the sublinear regret rate can be preserved against any linear policy by tuning ζ, which can be achieved by increasing the DAC memory parameter p (see Lemma 5.2). Next, we design algorithms that minimize $\mathcal{R}_T(\pi_{1,\dots,T}, \pi_\star)$.

5.3. NSC ORIGINS

The NSC thread was initiated in [178] which designed the first policy with sub-linear regret for dynamical systems, aspiring to generalize the classical control problem to general convex cost functions and arbitrary disturbances. In essence, the system state is modeled as a limited memory and the problem is cast into the standard OCO framework, which allows recovering the OCO bounds scaled by the memory length. Follow-up works refined these results for strongly convex functions [184, 185, 186]; and systems where the actions are subject to fixed or adversarially-changing constraints [187, 188]. NSC was also extended to systems where matrices (A, B) are unknown [85], systems with bandit feedback [189], and time-varying systems [190]. As expected, the regret bounds deteriorate in these cases, e.g., becoming $\mathcal{O}(T^{2/3})$ for unknown systems and $\mathcal{O}(T^{3/4})$ for systems with bandit feedback. Efficiency is also investigated in [191] with projection-free methods. All these works provide bounds that scale with the number of time steps T, as opposed to the adaptive and optimistic bounds presented here, which are proportional to the witnessed costs and perturbations, and prediction errors.

The NSC framework was also investigated using other metrics such as dynamic policy regret [192], adaptive policy regret [182], and competitive ratio [193, 194]. For dynamic and adaptive regret, methods with static regret guarantees, as discussed here, are used as building blocks for "meta" algorithms with static/adaptive guarantees [190, 195]. For competitive ratio, the problem model often has different semantics from the model considered here. For example, the adversary needs to reveal information about the cost function before the learner commits to a decision [193], or the cost is assumed to be a fixed quadratic function [194]. More generally, it was demonstrated in [196] that a regret guarantee against the optimal DAC policy automatically implies a competitive ratio with an additive sub-linear term. Hence, the presented algorithm is still highly relevant even for other metrics.

A related and recently refreshed line of work that also considers adversarial costs and transitions in stateful systems is Adversarial MDPs [197]. Unlike NSC, this line considers finite states and actions, whose sizes appear in the bounds. Thus, their

algorithms are irreducible to our settings.

5.4. Part I: Adaptive Online NSC

Adaptivity has been an important concept in OCO, and we refer the readers to the survey in [112], or the remarks in [28, Sec. 3.5] for details. In adaptive learning the regret bounds scale proportionally to the witnessed costs. Hence, for *easy* environments with small costs, the bounds are considerably tighter than the standard worst-case ones which assume maximum cost at each round. On the other hand, if the environment follows actually a worst-case scenario, i.e., the adversary induces costs with large gradients that fluctuate aggressively, the adaptive bounds remain sublinear but have worse constant factors. Indeed, there is a price to be paid for adaptivity. For policy regret, the only form of adaptive bounds appears in [182, Thm. 2], which, however, contain additive constants[4] (i.e., do not directly collapse to 0 even when all costs are 0). Additionally, the authors do not consider the DAC policy class but the actions are rather a result of a "betting" technique that is suited to the "tracking" sub-task in control. In fact, any DAC policy can be combined with their meta-algorithm to obtain guarantees on the "tracking", and hence our works are complementary (See discussion in [182, Sec 4.2]).

5

5.4.1. Methodology and Contributions

The analysis approach of NSC algorithms relies on approximating the cost of the non-stationary policy $\sum_{t=1}^{T} c_t(\boldsymbol{x}_t, \boldsymbol{u}_t)$ by the counterfactual stationary one $\sum_{t=1}^{T} c_t(\boldsymbol{x}_t(\pi_t), \boldsymbol{u}_t(\pi_t))$ with a sub-linear approximation term of order $\mathcal{O}(\sqrt{T})$. This is possible due to the structure of LTI systems, where the effect of far-in-the-past actions decays exponentially, deeming the cost similar to that of "OCO with bounded memory" framework [198]. Then, Online Gradient Descend (OGD) is used to obtain an $\mathcal{O}(\sqrt{T})$ bound for the stationary policy regret, leading to a total policy regret of the same order. In our case, however, we adopt the more general FTRL framework, cf. [112], which makes it no longer apparent that such a reduction is possible.

In detail, FTRL operates on the principle that at each t, the learner optimizes its decision[5] M_{t+1} by minimizing the aggregate cost until t, plus a strongly convex regularization term that penalizes the deviation from M_t. It is known that when these regularization terms are adaptive (i.e., the strong convexity is incremented at each t proportionally to the witnessed cost of M_t), we get bounds proportional to the observed costs. Now, the issue is that unlike GPC where, at each t, the distance between two previous decision variables M_s and M_{s+1}, $s \in [1, t-1]$ is (i) fixed and (ii) of order $\mathcal{O}(T^{-1/2})$, FTRL's adaptive regularizers make the distance between M_s and M_{s+1} (i) dependant on the costs witnessed until s (i.e, not fixed), and (ii) independent of the horizon T (or the time t). As these two properties were essential for proving GPC policy bounds, we are faced with a new challenge that is specific to FTRL and NSC integration. However, we show that the $c_t(\boldsymbol{x}_t, \boldsymbol{u}_t)$ cost can still approximate, up to a diminishing error, the counterfactual cost of $c_t(\boldsymbol{x}_t(\pi_t), \boldsymbol{u}_t(\pi_t))$.

[4] These works also target the "adaptive regret" metric.

[5] As will be detailed later, the action $\boldsymbol{u}_t$ is expressed as a linear combination of the parameters M.

This approximation effectively enables regularization based on the witnessed cost at t and eventually leads to the sought-after adaptive regret bound of the form $\mathcal{O}\big((\sum_{t=1}^{T} g_t)^{1/2}\big)$.

At a more conceptual level, it is interesting to observe that GPC and similar controllers already incorporated some notion of "adaptivity" in the sense that the performance guarantee is with respect to a *benchmark* that depends on the observed costs. This is clearly a more refined and less conservative approach than the classical H_∞ control framework that benchmarks itself against the worst-case scenario [199]. In this work, we make the next step, and not only use a cost-dependent benchmark like GPC, but also the performance of the controller w.r.t. that benchmark is itself shaped by the observed costs and perturbations. This is in contrast to GPC where the difference from the benchmark (i.e., the regret bound) does not depend on the encountered costs and perturbations. This additional layer of adaptability expedites the learning rates whenever the problem allows it (i.e., smaller or less volatile gradients are encountered). While the proposed adaptive controller is designed to benefit from easy environments, it does suffer performance degradation when the environment is indeed adversarial, but only in terms of constant factors. Hence, it provides a fail-safe tool for NSC.

5.4.2. AdaFTRL-C

We propose an algorithm (`AdaFTRL-C`) for optimizing the policy parameters M_t, $t \in [T]$. The algorithm uses the FTRL update formula:

$$M_{t+1} = \operatorname*{arg\,min}_{M \in \mathcal{M}} \left\{ \sum_{s=1}^{t} c_s(\boldsymbol{x}_s(\pi), \boldsymbol{u}_s(\pi)) + r_s(M) \right\}. \tag{5.7}$$

For the incremental regularizers $r_t(\cdot)$, we define[6] $g_t \doteq \max(\|G_t\|, \|G_t\|^2)$ and use:

$$r_t(M) = \frac{\sigma_t}{2}\|M - M_t\|^2, \text{ where } \sigma_1 = \sigma\sqrt{g_1},\ \sigma_t = \sigma\big(\sqrt{g_{1:t}} - \sqrt{g_{1:t-1}}\big). \tag{5.8}$$

Note that the sum of the counterfactual cost sequence in (5.7) is indeed a function of M since both $\boldsymbol{x}$ and $\boldsymbol{u}$ are written in terms of M as shown earlier. Defining the norm $\|\cdot\|_t \doteq \sqrt{\sigma_{1:t}}\|\cdot\|$ we get that the regularizer $r_t(\cdot)$ is 1- strongly convex w.r.t. norm $\|\cdot\|_t$, whose dual is $\|\cdot\|_{t,*} = \frac{1}{\sqrt{\sigma_{1:t}}}\|\cdot\|$.

Due to the cost linearization principle (see e.g., [180, Sec. 2.4]), (5.7) can be solved by two efficient steps: closed form solution of an unconstrained quadratic program, followed by a Euclidean projection, essentially matching the computational complexity of GPC.

Algorithm 11 summarizes the proposed routine; `AdaFTRL-C` first executes an action (line 3). Then, the cost function is revealed and G_t is recorded (line 4). The system then transitions to state $\boldsymbol{x}_{t+1}$, effectively revealing the disturbance vector $\boldsymbol{w}_t$ (lines 5, 6). The strong convexity parameter is calculated (line 7) and the next

[6] Unlike the discussion in the introduction, here we have g_t as the max of the norm and its square. The necessity of this originates from lower bounds on OCO with switching costs [200] and will become clear in the analysis.

action is committed through updating the policy parameters (line 8). The policy regret of this `AdaFTRL-C` routine is characterized by the following theorem:

Algorithm 11: Adaptive FTRL Controller (`AdaFTRL-C`)

Input: An intrinsically stable LTI system (A, B), $z \doteq pw\kappa_B$, $\sigma = \sqrt{\frac{\delta^2+2lz}{2\delta^2 k_M^2}}$, $M_1 \in \mathcal{M}$.
Output: $\{\boldsymbol{u}_t\}_{t=1}^T$

1 **for** $t = 1, 2, \ldots, T$ **do**
2 Use action $\boldsymbol{u}_t = \sum_{j=1}^p M_t^{[j]} \boldsymbol{w}_{t-j}$
3 Observe cost $c_t(\boldsymbol{x_t}, \boldsymbol{u_t})$ and record the gradient $G_t(M_t)$
4 Observe new state $\boldsymbol{x_{t+1}}$
5 Record the disturbance $\boldsymbol{w_t} = \boldsymbol{x}_{t+1} - A\boldsymbol{x_t} - B\boldsymbol{u_t}$
6 Update regularization parameters σ_t via (5.8)
7 Calculate M_{t+1} via (5.7)
end

5

Theorem 5.3. *Let (A, B) be an intrinsically stable system (i.e., $\|A\|_{op} \le 1 - \delta$) with $\|B\| \le \kappa_B$ and $\|\boldsymbol{w}\| \le w$. Let the cost $c_t(\cdot, \cdot)$ be l-Lipschitz, and assume that the DAC parametrization $M = [M^{[1]}, \ldots, M^{[p]}]$ is bounded: $\|M\| \le \kappa_M$. Define $z \doteq pw\kappa_B$ and $g_t \doteq \max(\|G_t\|, \|G_t\|^2)$. Algorithm `AdaFTRL-C` produces policies $\pi_{1,\ldots,T}$ such that for all T, the following bound holds:*

$$\mathcal{R}_T(\pi_{1,\ldots,T}, \pi_\star) \le \frac{2}{\delta}\sqrt{2\kappa_M^2\left(\delta^2 + 2lz\right)}\sqrt{\sum_{t=1}^T g_t} \quad .$$

Discussion. `AdaFTRL-C` achieves a policy regret that scales according to the *witnessed* costs' gradients. This can be much tighter than the bounds of non-adaptive controllers, like GPC, in easy environments. The improvement depends on how smaller the $\|G_t\|$ values are than g (an upper bound on $\|G_t\|$). In the worst case ($\|G_t\| = g, \forall t$), our bound is worse by a constant factor compared to GPC[7], but maintains its sub-linearity. This is expected since `AdaFTRL-C` is conservative in regularization. Hence, when the losses are large, its performance is slightly degraded.

Proof layout. First, we bound the difference between the learner's accumulated cost, which is induced by $\pi_{1,\ldots,t}$, and the counterfactual accumulated cost had the learner followed π_t in *all* steps $1, \ldots, t$ (i.e., stationary policy π_t). Second, since the cost c_t is convex in the parametrization M_t of such stationary policy, we leverage the FTRL theory to bound the regret against the parameters $M_\star$.

Lemma 5.4. *Let $c_t(\boldsymbol{x}_t, \boldsymbol{u}_t)$ be the cost induced by following $\pi_{1,\ldots,t}$, and $c_t\big(\boldsymbol{x}_t(\pi_t), \boldsymbol{u}_t(\pi_t)\big)$ the cost induced by following the policy $\pi_t, \forall \tau \le t$, where each π_t*

[7] Using OGD update rule instead would result in better *worst-case* bound by a factor of $\sqrt{2g}$.

is derived using (5.7). *Let* $z \doteq pw\kappa_B$ *Then:*

$$\nu_t \doteq \Big| c_t\left(\boldsymbol{x}_t, \boldsymbol{u}_t\right) - c_t\big(\boldsymbol{x}_t(\pi_t), \boldsymbol{u}_t(\pi_t)\big) \Big| \leq lz \sum_{i=0}^{t-1} (1-\delta)^i \sum_{\tau=t-i-1}^{t-1} \frac{g_\tau}{\sigma\sqrt{g_{1:\tau}}} \doteq \widehat{\nu}_t.$$

Proof. Since c_t is l-Lipschitz, it suffices to bound the distance between its arguments; and because it holds[8] $\boldsymbol{u}_t(\pi_{1,\dots,t}) = \boldsymbol{u}_t(\pi_t)$, we only need to bound $\|\boldsymbol{x}_t - \boldsymbol{x}_t(\pi_t)\|$. From Lemma 5.1:

$$\begin{aligned} \boldsymbol{x}_t &= \sum_{i=0}^{t-1} A^i \Bigg(B \sum_{j=1}^{p} M_{t-i-1}^{[j]} \boldsymbol{w}_{t-i-j-1} + \boldsymbol{w}_{t-i-1} \Bigg) \\ &= \sum_{i=0}^{t-1} A^i \big(B M_{t-i-1}\ \boldsymbol{w}_{t-i-1,p} + \boldsymbol{w}_{t-i-1} \big), \end{aligned} \tag{5.9}$$

where we defined $\boldsymbol{w}_{t-i,p} \doteq (\boldsymbol{w}_{t-i-1}, \dots, \boldsymbol{w}_{t-i-p})$ so as to express the vector $\sum_{j=1}^{p} M_t^{[j]}\ \boldsymbol{w}_{t-j}$ equivalently as $M_t\ \boldsymbol{w}_{t-1,p}$. Similarly, we can write:

$$\boldsymbol{x}_t(\pi_t) = \sum_{i=0}^{t-1} A^i \big(B M_t\ \boldsymbol{w}_{t-i-1,p} + \boldsymbol{w}_{t-i-1} \big). \tag{5.10}$$

Note that we have M_t in (5.10) instead of M_{t-i-1} in (5.9). This is because for the counterfactual state $\boldsymbol{x}_t$, $M_{t-i-1} = M_t, \forall i < t$. Now, subtracting the above two state expressions, we have: $\|\boldsymbol{x}_t - \boldsymbol{x}_t(\pi_t)\|$

$$\overset{(a)}{\leq} \sum_{i=0}^{t-1} \|A^i B (M_{t-i-1} - M_t)\|_{\mathrm{op}} \|\boldsymbol{w}_{t-i-1,p}\| \overset{(b)}{\leq} \sum_{i=0}^{t-1} \|A\|_{\mathrm{op}}^i \|B\|_{\mathrm{op}} \|(M_{t-i-1} - M_t)\|_{\mathrm{op}} \|\boldsymbol{w}_{t-i-1,p}\|$$

$$\overset{(c)}{\leq} \sum_{i=0}^{t-1} (1-\delta)^i \|B\| \|(M_{t-i-1} - M_t)\| \|\boldsymbol{w}_{t-i-1,p}\| \overset{(d)}{\leq} pw\kappa_B \sum_{i=0}^{t-1} (1-\delta)^i \|M_{t-i-1} - M_t\|$$

$$\overset{(e)}{=} z \sum_{i=0}^{t-1} (1-\delta)^i \|M_{t-i-1} - M_t\| \overset{(f)}{\leq} z \sum_{i=0}^{t-1} (1-\delta)^i \sum_{\tau=t-i-1}^{t-1} \frac{\|G_\tau\|}{\sigma_{1:\tau}}$$

$$\overset{(g)}{\leq} z \sum_{i=0}^{t-1} (1-\delta)^i \sum_{\tau=t-i-1}^{t-1} \frac{g_\tau}{\sigma\sqrt{g_{1:\tau}}},$$

where (a) follows by the triangular inequality and $\|A\boldsymbol{x}\| \leq \|A\|_{\mathrm{op}} \|\boldsymbol{x}\|$, (b) by the sub-multiplicative property of matrix norms, (c) by the bound on the spectral norm of A and $\|\cdot\|_{op} \leq \|\cdot\|$, (d) by $\|B\| \leq \kappa_B$, $\|\boldsymbol{w}_{t,p}\| \leq \sum_{b=1}^{p} \|\boldsymbol{w}_{t-b}\| = pw$, (e) by grouping $z \doteq pw\kappa_B$, (f) by the auxiliary Lemma 5.5, and finally (g) by the definition of g_t. Using l-Lipschitzness completes the proof. □

[8] Recall the stability with $K = 0$ assumption. Otherwise, action deviation is reducible to state deviation from (5.3).

Now that we have characterized the stationary vs. non-stationary cost deviation, we proceed into (i) bound the cost of the stationary policy (ii) bound this deviation term.

Proof. **of Theorem 5.3**. By the definition of ν_t (in Lemma 5.4), we can directly bound the policy regret:

$$\sum_{t=1}^{T}\Big(c_t(\boldsymbol{x}_t,\boldsymbol{u}_t)-c_t\big(\boldsymbol{x}_t(\pi_\star),\boldsymbol{u}_t(\pi_\star)\big)\Big)\leq\sum_{t=1}^{T}\Big(c_t\big(\boldsymbol{x}_t(\pi_t),\boldsymbol{u}_t(\pi_t)\big)-c_t\big(\boldsymbol{x}_t(\pi_\star),\boldsymbol{u}_t(\pi_\star)\big)\Big)+\nu_{1:T}$$

For the first sum, note that $c_t\left(\boldsymbol{x}_t(\pi_t),\boldsymbol{u}_t(\pi_t)\right)$ is a function of only the iterates M_t and recall that M_t are computed according to (5.7)). Hence, we recognize this sum as simply the standard FTRL regret, which can be bounded using the known bound [112, Thm. 2]. Thus,

$$\mathcal{R}_T(\pi_{1,\dots,T},\pi_\star)\leq\sum_{t=1}^{T}r_t(M_\star)+\frac{1}{2}\sum_{t=1}^{T}\big\|\nabla c_t\big(\boldsymbol{x}_t(\pi_t),\boldsymbol{u}_t(\pi_t)\big)\big\|_{t,*}^2\quad+\nu_{1:T}.$$

We proceed to bounding each of the three summations above.

$$(i):\sum_{t=1}^{T}r_t(M_\star)=\sum_{t=1}^{T}\frac{\sigma_t}{2}\|M_\star-M_t\|^2\leq 2\kappa_M^2\sum_{t=1}^{T}\sigma_t\overset{(a)}{\leq}2\sigma\kappa_M^2\sqrt{g_{1:T}},$$

$$(ii):\frac{1}{2}\sum_{t=1}^{T}\big\|\nabla c_t\big(\boldsymbol{x}_t(\pi_t),\boldsymbol{u}_t(\pi_t)\big)\big\|_{t,*}^2\leq\frac{1}{2}\sum_{t=1}^{T}\frac{\|G_t\|^2}{\sigma_{1:t}}\leq\frac{1}{2}\sum_{t=1}^{T}\frac{g_t}{\sigma\sqrt{g_{1:t}}}\overset{(b)}{\leq}\frac{1}{\sigma}\sqrt{g_{1:T}}$$

where inequality (a) is due to the telescoping sum of σ_t, and (b) is due to the common tool [201, Lemma 3.5], or its extended version [28, Lem. 4.13]). The $\nu_{1:T}$ term is especially challenging since the standard tools from OCO do not suffice. Namely, observe that we cannot use Auer's lemma again for $\widehat{\nu}_{1:T}$, since we are adding, at each t, t many terms, each divided by a different quantity. Hence, we prove a more general result in auxiliary Lemma 5.6, which, we believe, is of independent interest as it can be used to provide similar bounds in learning algorithms for systems with memory. With this new technical result at hand, we get:

$$(iii):\nu_{1:T}\overset{(c)}{\leq}\widehat{\nu}_{1:T}=lz\sum_{t=1}^{T}\sum_{i=0}^{t-1}(1-\delta)^i\sum_{\tau=t-i-1}^{t-1}\frac{g_\tau}{\sigma\sqrt{g_{1:\tau}}}\overset{(d)}{\leq}\frac{2lz}{\sigma\delta^2}\sqrt{g_{1:T}},$$

where (c) is from the definition of $\widehat{\nu}_t$ and (d) is due to using Lemma 5.6 with $f(x)=1/\sqrt{x}$, $a_t=g_t$, and $a_0=0$. Tuning σ we get $\sigma=\sqrt{\frac{\delta^2+2lz}{2\delta^2k_M^2}}$. Substituting in $(i),(ii),$ and (iii) we get the bound. □

Auxiliary Lemmas

Lemma 5.5. *The distance between parameters M_{t-i-1} and M_t, with $i \leq t-1$, where each M_t is updated using (5.7), with regularizers (5.8), can be bounded as* $\|M_{t-i-1} - M_t\| \leq \sum_{\tau=t-i-1}^{t-1} \frac{\|G_\tau\|}{\sigma_{1:\tau}}$.

Proof. First, note that $\|M_{t-i-1} - M_t\| \leq \sum_{\tau=t-i-1}^{t-1} \|M_\tau - M_{\tau+1}\|$. To bound each $\|M_\tau - M_{\tau+1}\|$, we leverage [112, Lem. 7] which states that given a convex function $\phi_1(\cdot)$ with a minimizer $q_1 \doteq \arg\min_q \phi_1(q)$ and a function $\phi_2(\cdot) = \phi_1(\cdot) + \psi(\cdot)$ that is a strongly convex w.r.t a norm $\|\cdot\|$ and has minimizer $q_2 \doteq \arg\min_q \phi_2(q)$, then we can bound the two minimizers as $\|q_1 - q_2\| \leq \|b\|_*$ for some (sub)gradient $b \in \partial\psi(q_1)$. Now, we invoke this result by setting:

$$q_1 = M_\tau, \quad q_2 = M_{\tau+1}, \quad \phi_1(M) = \sum_{s=1}^{\tau-1} \Big(c_s\big(\boldsymbol{x}_s(\pi), \boldsymbol{u}_s(\pi)\big) + r_s(M) \Big) + r_\tau(M), \quad \text{and}$$

$$\phi_2(M) = \sum_{s=1}^{\tau-1} \Big(c_s\big(\boldsymbol{x}_s(\pi), \boldsymbol{u}_s(\pi)\big) + r_s(M) \Big) + r_\tau(M) + \underbrace{c_\tau\big(\boldsymbol{x}_\tau(\pi), \boldsymbol{u}_\tau(\pi)\big)}_{\psi(M)}.$$

The function $\phi_2(M)$ is strongly-convex w.r.t. the norm $\|\cdot\|_\tau = \sqrt{\sigma_{1:\tau}}\|\cdot\|$, a property inherited due to containing the sum of all regularizers up to step τ, i.e., $\sum_{s=1}^{\tau} r_s(M)$. And the dual norm of the gradient of the above-defined $\psi(M)$ function, at each step s, is upper-bounded by $\|G_s\|_{\tau,*}$.

Finally, it suffices to observe that: (i) based on the definition of the update rule for variables M, and (ii) the fact that $r_t(M), \forall t$ is a proximal regularizer thus $r_t(M_t) = 0, \forall t$, we indeed have that $M_\tau = \arg\min_{M\in\mathcal{M}} \phi_1(M)$ and $M_{\tau+1} = \arg\min_{M\in\mathcal{M}} \phi_1(M) + \psi(M)$. Now, applying [112, Lem. 7] we obtain $\|M_\tau - M_{\tau+1}\|_\tau \leq \|G_\tau\|_{\tau,*} \rightarrow \|M_\tau - M_{\tau+1}\| \leq \frac{\|G_\tau\|}{\sigma_{1:\tau}}$. □

Lemma 5.6. *Let $a_\tau \geq 0\ \forall\tau$, $\delta \in (0,1]$ and $f : [0,\infty] \mapsto [0,\infty)$ be a non-increasing function. Then:*

$$\mathcal{S} \doteq \sum_{t=1}^{T} \sum_{i=0}^{t-1} (1-\delta)^i \sum_{\tau=t-i}^{t} a_\tau\ f(a_{0:\tau}) \leq \frac{1}{\delta^2} \int_{a_0}^{a_{1:T}} f(x)dx$$

Proof. Instead of summing over all $i < t$ for each $t \in [1,T]$, the sum $\mathcal{S}$ can be equivalently re-written by summing over all $t > i$ for each $i \in [0,T]$:

$$\mathcal{S} = \sum_{i=0}^{T} (1-\delta)^i \sum_{t=i+1}^{T} \sum_{\tau=t-i}^{t} a_\tau\ f(a_{0:\tau}) \tag{5.11}$$

Now we bound $\mathcal{S}' \doteq \sum_{t=i+1}^{T} \sum_{\tau=t-i}^{t} a_\tau\ f(a_{0:\tau})$. Let $s_t \doteq a_{0:t}$

$$\sum_{\tau=t-i}^{t} a_\tau\ f(a_{0:\tau}) \overset{(a)}{=} \sum_{\tau=0}^{i} a_{t-\tau} f(s_{t-\tau}) \overset{(b)}{=} \sum_{\tau=0}^{i} \int_{s_{t-\tau-1}}^{s_{t-\tau}} f(s_{t-\tau})dx \overset{(c)}{\leq} \sum_{\tau=0}^{i} \int_{s_{t-\tau-1}}^{s_{t-\tau}} f(x)dx.$$

$$\text{Hence, } \mathcal{S}' \leq \sum_{\tau=0}^{i} \sum_{t=i+1}^{T} \int_{s_{t-\tau-1}}^{s_{t-\tau}} f(x)dx \overset{(d)}{=} \sum_{\tau=0}^{i} \int_{s_{i-\tau}}^{s_{T-\tau}} f(x)dx$$

$$\overset{(e)}{=} \sum_{\tau=0}^{i} \int_{s_0}^{s_T} f(x)dx = (i+1) \int_{s_0}^{s_T} f(x)dx.$$

Where (a) follows by changing the sum index, (b) is from $\int_n^m c\ dx = (m-n)c$, $\forall n, m, c \in \mathbb{R}$, (c) from $f(\cdot)$ being non-increasing, (d) from the additive property of integrals, and (e) from expanding the integration limits and positivity of $f(\cdot)$. Substituting the bound for $\mathcal{S}'$ back in (5.11) we get:

$$\mathcal{S} \leq \sum_{i=0}^{T} (1-\delta)^i (i+1) \int_{s_0}^{s_T} f(x)dx \leq \frac{1}{\delta^2} \int_{s_0}^{s_T} f(x)dx.$$

Where we used that $\sum_{i=0}^{\infty} i(1-\delta)^i \leq {}^{1-\delta}/_{\delta^2}$ and $\sum_{i=0}^{\infty}(1-\delta)^i \leq {}^{1}/_{\delta}$. □

5

5.4.3. Numerical examples

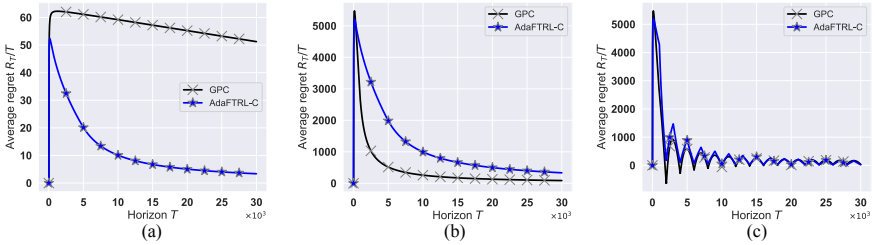


Figure 5.1: Average regret when (a): $\boldsymbol{\theta}_t = (1,1), \boldsymbol{w}_t = -(0.1,0.1)\forall t$; (b): $\boldsymbol{\theta}_t = (10,10), \boldsymbol{w}_t = -(1,1)\forall t$; (c): $\boldsymbol{\theta}_t = (10,10)$ or $-(10,10)$ (Alternating every 1000 steps), $\boldsymbol{w}_t = -(1,1)\forall t$.

We conclude with numerical examples of the implications of adaptivity in NSC. We consider an LTI system with $\boldsymbol{x}, \boldsymbol{u} \in \mathbb{R}^2$, $p = 10$, and hence $M \in \mathbb{R}^{2\times 20}$. The dynamics are $A = 0.9 \times \boldsymbol{I}_2, B = \boldsymbol{I}_2$, with perturbation of maximum magnitude of $w = \sqrt{2}$. We consider a linear cost $c_t = \langle \boldsymbol{\theta}_t, \boldsymbol{x}_t \rangle$, with $\boldsymbol{\theta}_t \in [-10, 10]^2$ and hence $l = \sqrt{200}$. Note that for $G_t^{(i,j)}$, the coefficient of $M_t^{(i,j)}$ when $c_t(x_t(\pi), u_t(\pi))$ is written explicitly in terms of M, we have that $G_t^{(i,j)} \leq \theta_t^{(i)} \times {w_t^j}/{0.1} = 100$ and hence $g = 633$. $\boldsymbol{\theta}_t$ and $\boldsymbol{w}_t$ are set according to different scenarios in Fig. 5.1. In scenario (a), the encountered environment ($\boldsymbol{\theta}_t$ and $\boldsymbol{w}_t$) are smaller than the worst-case costs (by a factor of 10). `AdaFTRL-C` takes advantage of this and accelerates the optimization, leading to an improved average regret by roughly the same order. In fact, (a) represents a whole class of easy environments where the witnessed costs are smaller than their maximum values, and in all these cases, `AdaFTRL-C` outperforms GPC. In the worst-case scenario (b), the degradation of `AdaFTRL-C` reaches a maximum of only 3.9 times that of GPC. Lastly, (c) serves to demonstrate that even in highly volatile environments and worst-case costs/perturbation, `AdaFTRL-C` closely matches the performance of GPC. Concluding, we see that adaptivity offers a lot of potential

gains for an entire family of easy environments, with tolerable degradation in the (single) worst-case scenario. The implementation of `AdaFTRL-C` and the code to reproduce the experiments are available in the repository [202].

5.5. Part II: Optimistic Online NSC

In contrast to the typical NSC framework, here we consider the existence of a *prediction oracle*. This oracle provides the learner, before committing to the action $\boldsymbol{u}_t$, with a forecast for the as-yet-unobserved cost functions $\{c_\tau(\cdot,\cdot)\}_{\tau=t}^{t+d}$ for a specific horizon of d slots. We denote the oracle's outcome as $\tilde{c}_\tau(\cdot,\cdot)$. Notably, the predictions themselves can be influenced by the adversary, meaning that no assumptions on the oracle's *accuracy* are made; the forecasted parameters may either deviate arbitrarily from the truth or be actually accurate in the best case.

While optimism has been studied for stochastic predictions [203, 204], and adversarial predictions [26, 150], these findings have not been applied to dynamical systems. In dynamical systems, the study of predictions is limited to either *perfect* predictions [205, 206], or *fixed* quadratic (thus, strongly convex) cost functions [207, 208, 209]. Our paper contributes to filling this gap. Predictions may also be viewed via the lens of "context" [210] in the problem of stochastic MDPs with *finite* states and actions. Lastly, the previous works of [208, 209] consider that the full predictions are provided *a priori*, meaning that the learner cannot benefit from updated predictions. Like MPC, we drop this assumption and allow the learner to use the most recent predictions.

This part is also related to the robust MPC literature [211, 212], where (possibly inaccurate) predictions are used. However, the main difference is that while our algorithm is robust to bad predictions, it is not designed based on them. Specifically, our benchmark policy changes when worst case costs and predictions are not witnessed (as dictated by the regret metric). A newer line of research studies the MPC-style algorithms under the regret metric [213, 214]. Perfect predictions are assumed in [213], whose bound was later generalized in [214] with parameterized predictions error. While these works consider the refined dynamic regret metric, their regret scales linearly with the prediction error.

Predictions in NSC. The prediction model we use has several advantages over those that appear in the preceding section. (*i*) We allow the prediction oracle to update its forecast *at every decision slot* t (similar to MPC). This flexibility is important, as in practice predictions can improve with time. (*ii*) the analysis reveals that the parameter d needs to scale logarithmically with T. This implies that predictions are not required for the entire future. (*iii*) the presented algorithm and its guarantees place no assumptions on the predictions' quality. To our knowledge, this represents the most general prediction-based setting for online control.

5.5.1. Motivation

The motivation for this addition on the NSC framework stems from the abundance of machine learning forecasting models, which provide high potential improvement if they are adequately accurate. The effect of such predictions on the classical re-

gret metric (not policy regret) has been studied in the literature of optimistic online learning, where the objective is to provide guarantees that scale with the accuracy of predictions while always staying sub-linear. Namely: $\mathcal{R}_T = \mathcal{O}(1)$ when all predictions are accurate and $\mathcal{O}(\sqrt{T})$ in all cases. That is, we are assured of reaching the performance of the benchmark policy, yet at a significantly improved rate when predictions happen to be precise. Hence, optimistic online learning represents a desirable combination of the best of both worlds: optimal worst-case guarantees with achievable best case guarantees. In fact, optimistic learning algorithms have been attracting considerable attention as the driving force behind recent state-of-the-art results in online constrained optimization [132], online discrete optimization [150], online sub-modular optimization [215], and online fairness [152] to name a few.

Unfortunately, such optimistic algorithms are yet to find their way to online NSC. This might be surprising given that previous NSC results were obtained through a streamlined reduction to the standard Online Convex Optimization (OCO) framework [216]. Hence, one might anticipate that optimistic NSC algorithms can be derived using a similar approach. Interestingly, this is not the case. Combining optimistic learning and NSC poses unique challenges for both frameworks. **First**, existing optimistic learning algorithms do not consider cost functions *with memory* and hence cannot handle states. Particularly, these algorithms update their regularizers at each time slot based on the accuracy of the prediction used by the preceding action [25, 26]. However, in stateful systems, an action made at t will have an effect that spans across all slots until $t + d$, and thus uses predictions for all these slots. Since the accuracy of such multi-step predictions is not available until $t + d + 1$, we cannot update the regularizer in the standard way. **Second**, the guarantee of NSC is established via a reduction to the OCO with Memory (OCO-M) framework [179], which in turn is reduced to the standard OCO [180] via the concept of slowly moving decision variables [178, Thm. 4.6] [179, Thm. 3.1]. This later reduction cannot be utilized in optimistic learning where accurate predictions lead to little or *no* regularization, driving consecutive decisions of the optimistic algorithm to vary arbitrarily (up to the set diameter) [26, Sec 2.2], [27, Sec 7.4].

This part tackles exactly these challenges and aims to answer the question: *is it possible to design an online learning algorithm whose policy regret is commensurate with the accuracy of an exogenous prediction oracle, while always staying sub-linear?* We answer this positively and builds upon recent advances in online learning, introducing, to our knowledge, the first optimistic controller for NSC.

5.5.2. Contributions

We achieve such optimistic guarantees by departing from the standard analysis approach of reducing the learner's non-stationary policy to a stationary one [178, 191, 192], and instead directly analyzing the non-stationary policy. Specifically, to address the first challenge, we demonstrate that the additive separability of the linearized costs allows expressing the costs as a sum of *memoryless* but *delayed* functions of each of the decision variables. Next, to tackle the second challenge, we analyze the performance of each decision variable *separately* via an alternative reduction to the framework of "optimism with delay" [86]. Nonetheless, we cus-

tomize this later framework with a specific "hint" design that exploits the structure of NSC where the cost is indeed delayed but still *gradually* being revealed at each step, leading to tighter bounds. We make these intuition-focused points concrete in the upcoming analysis.

The main contribution is an optimistic controller with policy regret scaling from $\mathcal{O}(1)$ to $\mathcal{O}(\sqrt{T})$, based on prediction accuracy. The methodology hinges on a new perspective on stateful systems (with a prediction oracle) as systems with delayed feedback, for which we build upon recent results on delay and optimism.

5.5.3. Online Control with Optimistic FTRL

Technical Assumptions

Assumption 1. *The system (A, B) is intrinsically stable: $\|A\|_{op} \leq 1 - \delta$ for some $\delta \in (0, 1]$.*

The above assumption allows us to satisfy the strong stability assumption with K being the zero matrix. Otherwise, we can revert to the strong stability assumption itself. This simplification facilitates the analysis without much loss in generality, as discussed in, for example, [182, Remark 4.1]. Additionally, we assume $\|B\| \leq \kappa_B$. The boundedness of $\|A\|$ follows from its spectral norm bound.

Recall (from the previous part) that when using DAC policies, the state at $t+1$ can be described as a linear transformation of the parameters chosen by the learner in the previous t slots $M_1, M_2, \ldots, M_t$ as in (5.4), which we restate here with a slightly different notation to facilitate the presentation in this part.

$$\begin{aligned} \boldsymbol{x}_{t+1} &= \sum_{i=0}^{t} A^i \left(B \sum_{j=1}^{p} \left(M_{t-i}^{[j]} \, \boldsymbol{w}_{t-i-j} \right) + \boldsymbol{w}_{t-i} \right) \\ &= \sum_{i=0}^{t} A^i \left(B M_{t-i} \, \overline{\boldsymbol{w}}_{t-i-1} + \boldsymbol{w}_{t-i} \right), \end{aligned} \tag{5.12}$$

where we defined $\overline{\boldsymbol{w}}_{t-i-1} \doteq (\boldsymbol{w}_{t-i-1}, \ldots, \boldsymbol{w}_{t-i-p})$ so as to express the vector $\sum_{j=1}^{p} M_t^{[j]} \boldsymbol{w}_{t-j}$ compactly as $M_t \overline{\boldsymbol{w}}_{t-1}$. The above expression for the state can be obtained by simply unrolling the dynamic in (5.1) and assuming, w.l.o.g that the initial state $\boldsymbol{x}_1$ (before executing $M_1, \ldots, M_t$) is $\mathbf{0}$. This is proven, e.g., in [82, Lem. 7.3] and [182, Lem. 4.3].

Cost functions. While our guarantees still hold for general convex cost functions, we focus here on the linear case:

Assumption 2. *The cost is linear in the state and action $c_t(\boldsymbol{x}_t, \boldsymbol{u}_t) = \langle \boldsymbol{\alpha}_t, \boldsymbol{x}_t \rangle + \langle \boldsymbol{\beta}_t, \boldsymbol{u}_t \rangle$. $\|\boldsymbol{\alpha}_t\| \leq \alpha$ and $\|\boldsymbol{\beta}\| \leq \beta$.*

The linearity assumption provides a useful structure in the analysis (separability) and enables us to quantify the prediction error in terms of the parameters $\boldsymbol{\alpha}_t$ and $\boldsymbol{\beta}_t$. It does not, however, compromise the presented regret guarantees. In fact, the linear costs are the most challenging[9] in the online learning settings; the regret

[9] As indicated in Sec. II, in the case of strongly convex costs, a tighter regret bound of $\mathcal{O}(\log(T))$ (compared to $\mathcal{O}(\sqrt{T})$) is possible [186].

caused by a sequence of general convex function can be indeed upper bounded by the regret caused by linearization of those functions. Thus, online learning works often focused on the linear case (see discussion on linearization in [180, Sec. 2.4]). Future costs can thus be predicted through the oracle's output $\{\boldsymbol{\alpha}_\tau, \boldsymbol{\beta}_\tau\}_{\tau=t}^{t+d}$.

OptFTRL-C

We introduce first few definitions to facilitate the presentation of the algorithm. Define the *forward cost* function:

$$F_t(M) \doteq \sum_{i=0}^{d} f_{t+i}^{(i)}(M),$$

where each $f_{t+i}^{(i)}(M)$ function shall describe the contribution of M to the cost experienced at slot $t+i$, and is defined as

$$f_t^{(i)}(M) \doteq \begin{cases} \langle \boldsymbol{\alpha}_t, \boldsymbol{\psi}_t^{i-1}(M) \rangle & \text{if } i \geq 1, \\ \langle \boldsymbol{\beta}_t, \boldsymbol{\psi}_t(M) \rangle & \text{if } i = 0, \end{cases} \tag{5.13}$$

with $\boldsymbol{\psi}_t^i : \mathbb{R}^{d_u \times (d_x p)} \mapsto \mathbb{R}^{d_x}$, and $\boldsymbol{\psi}_t : \mathbb{R}^{d_u \times (d_x p)} \mapsto \mathbb{R}^{d_u}$ being the following linear transformations, which are used to simplify the presentation of the action and state expression in (5.3) and (5.12), respectively:

$$\boldsymbol{\psi}_{t+1}^i(M) \doteq A^i(BM\overline{\boldsymbol{w}}_{t-i-1} + \boldsymbol{w}_{t-i}), \tag{5.14}$$

$$\boldsymbol{\psi}_t(M) \doteq M\overline{\boldsymbol{w}}_{t-1}. \tag{5.15}$$

The role of the functions in (5.13) will become clear later in the analysis. Roughly, the cost c_t will be expressed as a sum of them. Denoting by $G_t^{(i)} = \nabla_M f_t^{(i)}(M)$, we have:

$$G_t^{(i)} = \begin{cases} B^\top (A^{i-1})^\top \boldsymbol{\alpha}_t \overline{\boldsymbol{w}}_{t-i-2}^\top & \text{if } i \geq 1, \\ \boldsymbol{\beta}_t \overline{\boldsymbol{w}}_{t-1}^\top & \text{if } i = 0. \end{cases} \tag{5.16}$$

Note that $G_t^{(i)}$ is a $d_u \times d_x p$ matrix with the (m,n)-th element being the partial derivative of $f_t^{(i)}$ w.r.t. the (m,n)-th element of M. From the above, we can get the bounds

$$\|G_t^{(i)}\| \leq \begin{cases} \alpha \kappa_B p w (1-\delta)^{i-1} & \doteq g^{(i)} \quad \text{if } i \geq 1, \\ \beta p w & \doteq g^{(0)} \quad \text{if } i = 0, \end{cases} \tag{5.17}$$

using that for any matrices A,B, and vector $\boldsymbol{w}$ $\|AB\| = \|A\|\|B\|$ and $\|A\boldsymbol{w}\| \leq \|A\|_{\text{op}}\|\boldsymbol{w}\|$. We also define the prediction $\tilde{G}_t^{(i)}$, which we can construct by plugging the oracle's output in (5.16), and hence we have the *partial* prediction error:

$$\Delta_t^{(i)} \doteq \|G_t^{(i)} - \tilde{G}_t^{(i)}\| \tag{5.18}$$

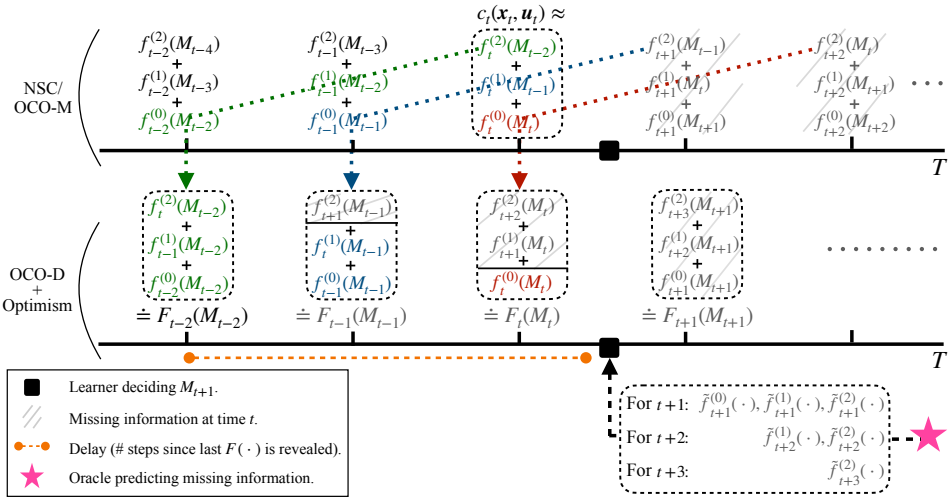


Figure 5.2: The methodology of designing `OptFTRL-C`. Up: The NSC to OCO-M reduction, with parameter $d = 2$ (detailed under subsection "NSC with linearized costs is separable OCO-M" in Sec. 8). Down: an equivalent *delayed OCO* formulation, obtained via rearrangement, which we append with an oracle (detailed in subsection "separable OCO-M is OCO-D" under Sec. 8).

We highlight that the prediction error magnitude decreases exponentially with i: $\Delta_t^{(i)} \propto (1-\delta)^i \leq e^{-i}$. This follows from (5.17) and the fact that predicting $\tilde{G}_t^{(i)}, i \geq 1$ amounts to predicting $\boldsymbol{\alpha}_t$ and attenuating it by a multiplication with A^{i-1}. We refer to i therefore as the attenuation level.

Similarly, we define G_t as the matrix of partial derivatives of $F_t(\cdot)$, and $\tilde{G}_t$ as its prediction. Lastly, we define the hybrid hint matrix H_t, which aims to approximate the sum $G_{t-d:t}$

$$H_t \doteq \underbrace{\sum_{i=0}^{d-1} \left(\sum_{j=0}^{d-i-1} G_{t-d+i+j}^{(j)} \right)}_{\text{observed at } t} + \underbrace{\sum_{j'=d-i}^{d} \tilde{G}_{t-d+i+j'}^{(j')} \mathbb{1}_{d \geq 1}}_{\text{future predictions}} + \tilde{G}_t, \tag{5.19}$$

and we denote by Δ_t the prediction error $\Delta_t \doteq \|G_{t-d:t} - H_t\|$. Due to the definition of $F_t(\cdot)$, certain elements of the summands (in $G_{t-d:t}$) are partially observed at t and are hence directly used in constructing H_t. The remaining elements in the sum are obtained from the prediction oracle.

With these definitions at hand, we can now introduce the main algorithmic step. We propose an algorithm (`OptFTRL-C`) for optimizing the policy parameters M_t, $t \in [T]$. The algorithm uses the update formula:

$$M_{t+1} = \arg\min_{M \in \mathcal{M}} \{ \langle G_{1:t-d} + H_{t+1}, M \rangle + r_{t+1}(M) \}, \tag{5.20}$$

where $r_t(\cdot)$ are strongly convex regularizers defined as:

$$r_{t+1}(M) = \frac{\lambda_{t+1}}{2}\|M\|^2,$$

$$\lambda_{t+1} = \frac{4}{\kappa_M} \max_{j \le t-d-1} \Delta_{j-d+1:j} + \frac{\sqrt{5}}{\kappa_M} \sqrt{\sum_{i=1}^{t-d} \Delta_i^2}. \tag{5.21}$$

In essence, the regularizer is a λ_{t+1}-strongly convex function, where λ_{t+1} is proportional to the observed prediction error up to t. At each t, we set the strong convexity to be the maximum sum of d consecutive witnessed errors, added to the root of the accumulated squared witnessed errors. This later term is aligned with memoryless OFTRL, whereas the former is necessary to adjust for the memory/delay effect.

Algorithm 12: `OptFTRL-C`

Input: System (A, B), parameter d, DAC parameters κ_M, p.
Output: Actions $\boldsymbol{u}_t$ at each slot $t = 1, \ldots, T$.

1 **for** *each time slot* $t = 1, \ldots, T$ **do**
2 Use action $\boldsymbol{u}_t = \sum_{j=1}^{p} M_t^{[j]} \boldsymbol{w}_{t-j}$
3 Observe $c_t(\boldsymbol{x_t}, \boldsymbol{u_t})$ and record the gradient G_{t-d}
4 Observe new state $\boldsymbol{x_{t+1}}$ and record $\boldsymbol{w}_t$;
5 Calculate Δ_{t-d} and update parameter λ_{t+1} via (5.21)
6 Receive future predictions $\{c_\tau(\cdot,\cdot)\}_{\tau=t+1}^{t+d}$
7 Construct H_{t+1} as in (5.19)
8 Calculate M_{t+1} via (5.20)
end

The steps of the `OptFTRL-C` routine are outlined in Algorithm 12. `OptFTRL-C` first executes an action $\boldsymbol{u}_t$ (line 2). Then, the cost function is revealed, completing the necessary information to compute G_{t-d} (line 3, recall that all the costs from $t-d, \ldots, t$ are required to know G_{t-d}). The system then transitions to state $\boldsymbol{x}_{t+1}$, effectively revealing the disturbance vector $\boldsymbol{w}_t$ (line 4). At this point, we can calculate Δ_{t-d} (since we know the ground truth G_{t-d}) and update the strong convexity parameter accordingly (line 5). Then, the oracle forecasts the next d costs functions (line 6), enabling us to construct the hybrid hint matrix (line 7). Finally, the next action $\boldsymbol{u}_{t+1}$ is committed through updating the policy parameters (line 8). The regret of this routine is characterized in the following theorem:

Theorem 5.7. *Let (A, B) be an LTI system, and $\{c_t(\cdot,\cdot)\}_{t=1}^T$, $\{\boldsymbol{w}\}_{t=1}^T$ be any sequence of stage costs and disturbances, respectively. Let $\Delta_t^{(i)}$ be the prediction error at t with attenuation level i, as defined in (5.18). Then, with memory parameter d defined as in Lemma 5.8, and under Assumptions 1 and 2, algorithm `OptFTRL-C` produces actions $\{\boldsymbol{u}_t\}_{t=1}^T$ such that for all T, the following holds:*

$$\mathcal{R}_T = \mathcal{O}\left(\sqrt{\sum_{t=1}^{T}\left(\sum_{i=0}^{d}\sum_{j=i}^{d}\Delta_{t+i}^{(d-j+i)}\right)^2}\right).$$

Discussion. `OptFTRL-C` achieves the sought-after accuracy-modulated bound that holds for systems with memory. It generalizes previous optimistic online learning bounds by incorporating memory (hence states), and generalizes previous online non-stochastic control bounds by handling predictions of unknown quality. Namely, `OptFTRL-C`'s bound has the following characteristics.

Prediction-commensurate: in the *best* case predictions ($\Delta_t^{(i)} = 0, \forall t$), the bound collapses to $\mathcal{O}(1)$, which is *constant*. On ther other hand, in the *worst* case, we get

$$\sum_{i=0}^{d}\sum_{j=i}^{d}\Delta_{t+i}^{(d-j+i)} \le 2\sum_{i=0}^{d}\sum_{j=i}^{d} g^{(d-j+i)} = 2\sum_{i=0}^{d}\sum_{k=i}^{d} g^{(k)} =$$
$$2\sum_{k=0}^{d}\sum_{i=0}^{k} g^{(k)} = 2\sum_{k=0}^{d}(k+1)g^{(k)} = 2g^{(0)} + 2\sum_{k=1}^{d}(k+1)g^{(k)}$$
$$\le 2\beta pw + \frac{2\alpha\kappa_B pw}{\delta^2} \doteq m$$

where the first equality follows from the triangular inequality and (5.17), and in the last inequality we used $\sum_{i=0}^{\infty} i(1-\delta)^i \le {}^{1-\delta}/_{\delta^2}$ and $\sum_{i=0}^{\infty}(1-\delta)^i \le {}^{1}/_{\delta}$. Hence, the regret becomes $\mathcal{O}(m\sqrt{T})$, which is order-optimal in T [28, Sec 5.1], achieving the optimistic premise.

Memory-commensurate: Apart from prediction adaptivity, `OptFTRL-C`'s performance is interpretable with respect to the spectrum of stateless to stateful systems. Consider the stateless case ($\boldsymbol{x}_t = \boldsymbol{0}, \forall t$). Hence, $c_t(\boldsymbol{x}_t, \boldsymbol{u}_t) = \langle \boldsymbol{\beta}_t, \boldsymbol{u}_t \rangle$. In this case, `OptFTRL-C` requires predictions only for the next step (as per (5.19)), The resulting bound becomes $\mathcal{O}((\sum_{t=1}^{T}\Delta_t^0)^{1/2}) = \mathcal{O}((\sum_t \|\boldsymbol{\beta}_t - \tilde{\boldsymbol{\beta}}\|)^{1/2})$, recovering the optimistic bound for *stateless* online learning [26].

On the other hand, consider a system with a general memory d. Then, `OptFTRL-C` uses predictions not only for the next step, but for the next $d+1$ steps $\{\boldsymbol{\beta}_\tau, \boldsymbol{\alpha}_t\}_{\tau=t}^{t+d}$, as dictated by (5.19). However, the dependence on future predictions' accuracy decays exponentially (recall that $\Delta_t^{(i)} \propto (1-\delta)^i$). For example, when $d = 1$ the resulting bound is $\mathcal{O}\big((\sum_{t=1}^{T}\Delta_t^{(0)} + \Delta_t^{(1)} + \Delta_{t+1}^{(1)})^{1/2}\big)$. I.e., we pay for the error in $d+1$ predictions[10], but with an exponentially decaying rate.

We now present the tools to prove Theorem 5.7. Our proof is structured into two primary parts. First, we demonstrate that the regret in linearized NSC is a specific instance within the OCO-M framework, achieved through a particular selection of separable functions (sub-section 8, visualized in Fig. 5.2 (up)). Second, we establish that the regret of OCO-M with these separable functions is in turn a particular case within the Delayed OCO (OCO-D) framework with a specific structure of delay (sub-section 8, visualized in Fig. 5.2 (down)). While the first part is fairly standard in NSC, we do not reduce its resulting OCO-M instance to standard OCO, but to OCO-D instead. The connection to delayed online learning was indeed identified in [186], but incorporating predictions (of unknown quality) was not considered.

[10]The fact that earlier errors get repeated is due to the compounding effect.

NSC with linearized costs is separable OCO-M.

In this subsection, we show how the cost at each time slot t can be approximated by the sum of a finite number of functions of only the past d decisions. Formally:

Lemma 5.8. *Given* $d \geq \frac{1}{\delta}\log(\frac{z}{\delta\epsilon}T)$, $z \doteq w(\kappa_B\kappa_M p+1)$. *Then, under Assumptions* 1&2, *for any* $\epsilon > 0$*:*

$$\sum_{t=1}^{T}\left|c_t(\boldsymbol{x}_t,\boldsymbol{u}_t) - \sum_{i=0}^{d} f_t^{(i)}(M_{t-i})\right| \leq \alpha\epsilon.$$

Proof. Define the counterfactual state $\hat{\boldsymbol{x}}_t$ as the state reached starting from $\mathbf{0}$ and then executing d DAC actions based on policies $M_{t-d}, M_{t-d+1}, \ldots, M_{t-1}$. In other words, this is the state reached at t by following the learner's policies but assuming that $\boldsymbol{x}_{t-d-1}$ was $\mathbf{0}$. From (5.12):

$$\hat{\boldsymbol{x}}_t = \sum_{i=0}^{d-1} A^i\big(BM_{t-i-1}\,\overline{\boldsymbol{w}}_{t-i-2} + \boldsymbol{w}_{t-i-1}\big). \tag{5.22}$$

For now, assume that $\|\hat{\boldsymbol{x}}_t - \boldsymbol{x}_t\| \leq \frac{\epsilon}{T}$. Then, by Assump. 2:

$$\sum_{t=1}^{T}|c_t(\boldsymbol{x}_t,\boldsymbol{u}_t) - c_t(\hat{\boldsymbol{x}}_t,\boldsymbol{u}_t)| \leq \|\boldsymbol{\alpha}_t\|\;\|\hat{\boldsymbol{x}}_t - \boldsymbol{x}_t\| = \alpha\epsilon.$$

However, $c_t(\hat{\boldsymbol{x}}_t,\boldsymbol{u}_t)$ can be written in terms of $f_t^{(i)}(\cdot)$ using again Assump. 2 but with the definitions in (5.14) and (5.15) :

$$\begin{aligned}
c_t(\hat{\boldsymbol{x}}_t,\boldsymbol{u}_t) &= \langle\boldsymbol{\alpha}_t, \sum_{i=0}^{d-1}\boldsymbol{\psi}_t^i(M_{t-i-1})\rangle + \langle\boldsymbol{\beta}_t,\boldsymbol{\psi}_t(M_t)\rangle \\
&= \sum_{i=0}^{d-1}\langle\boldsymbol{\alpha}_t,\boldsymbol{\psi}_t^i(M_{t-i-1})\rangle + \langle\boldsymbol{\beta}_t,\boldsymbol{\psi}_t(M_t)\rangle \\
&= \sum_{i=1}^{d}\langle\boldsymbol{\alpha}_t,\boldsymbol{\psi}_t^{i-1}(M_{t-i})\rangle + \langle\boldsymbol{\beta}_t,\boldsymbol{\psi}_t(M_t)\rangle = \sum_{i=1}^{d} f_t^{(i)}(M_{t-i}) + \langle\boldsymbol{\beta}_t,\boldsymbol{\psi}_t(M_t)\rangle \\
&= \sum_{i=0}^{d} f_t^{(i)}(M_{t-i}).
\end{aligned}$$

It remains to show that $\|\boldsymbol{x}_t - \hat{\boldsymbol{x}}_t\| \leq \frac{\epsilon}{T}$. From (5.12) and (5.22):

$$\begin{aligned}
\|\hat{\boldsymbol{x}}_t - \boldsymbol{x}_t\| &\leq \|\sum_{i=d}^{t-1} A^i\big(BM_{t-i-1}\overline{\boldsymbol{w}}_{t-i-2} + \boldsymbol{w}_{t-i-1}\big)\| \\
&\leq \sum_{i=d}^{t-1}(\|A^iBM\|_{op}\|\overline{\boldsymbol{w}}_{t-i-2}\| + \|A^i\|_{op}\|\boldsymbol{w}_{t-i-1}\|)
\end{aligned}$$

$$\overset{(a)}{\leq} \sum_{i=d}^{t-1} \|A^i\|_{op}\|BM\|\|\overline{\boldsymbol{w}}_{t-i-2}\| + \|A^i\|_{op}\|\boldsymbol{w}_{t-i-1}\| \overset{(b)}{\leq} \sum_{i=d}^{t-1}(1-\delta)^i w(\kappa_B\kappa_M p+1)$$
$$\overset{(c)}{\leq} z\int_{i=d}^{\infty} e^{-\delta i}di = \frac{z}{\delta}e^{-\delta d},$$

where (a) follow from the sub-multipilicitive property of $\|\cdot\|_{\text{op}}$, and $\|B\|_{\text{op}} \leq \|B\|$, (b) by Assump. 1, and the bounds on matrix norms, and (c) from $1-x \leq e^{-x}$ and the definition of z. Substituting d, makes the last term $\frac{\epsilon}{T}$. □

Thus, the closeness of $\boldsymbol{x}_t$ and $\hat{\boldsymbol{x}}_t$ is due to the (strong) stability. Lastly, we note the possibility to set d adaptively without knowledge of T using $d = \frac{1}{\delta}\log(\frac{zt}{\delta\epsilon})$.[11]

SEPARABLE OCO-M IS OCO-D

Now that we have approximated the cost at t by a separable function with memory, we show in this subsection that the separated functions can be rearranged to represent an equivalent OCO with delayed feedback formulation.

Lemma 5.9. *Let $f_t(M_0,\ldots,M_d)$ be a separable function: $f_t(M_0,\ldots,M_d) = \sum_{i=0}^{d} f_{t,i}(M_{d-i})$, $f_{t,i}(M) : \mathbb{R}^{(d_u\times d_x p)} \mapsto \mathbb{R}$. Let $\mathcal{A}$ be an online learning algorithm whose decisions M_{t+1} depend on the history set $\mathbb{H}_t \doteq \cup_{i=0}^{d}\{f_{\tau,i}(\cdot)\}_{\tau=1}^{t}$. Define $J_t(M) \doteq \sum_{i=0}^{d} f_{t+i,i}(M)$, and the history set w.r.t. $J_t(\cdot)$ as $\mathbb{H}_t^J$. Then,*

$$\mathbb{H}_t^J = \{J_\tau(\cdot)\}_{\tau=1}^{t-d}, \text{ and} \tag{5.23}$$
$$\sum_{t=1}^{T} f_t(M_{t-d},\ldots,M_t) = \sum_{t=1}^{T} J_t(M_t). \tag{5.24}$$

(5.24) *means that the accumulated cost, with memory, is equivalent to that of the memoryless functions $J_t(\cdot)$. However, from (5.23) $\mathcal{A}$ has delayed feedback w.r.t. $J_t(\cdot)$; when deciding M_{t+1}, feedback up to only $t-d$ is available.*

Proof. the first part is immediate from the definition of $J_\tau(\cdot)$; any $J_\tau(\cdot)$ with $\tau > t-d$ would require a function that is not in the original history set $\mathbb{H}_t$ (not known at t).

The second part is mainly index manipulation:

$$\sum_{t=1}^{T} f_t(M_{t-d},\ldots,M_t) = \sum_{t=1}^{T}\sum_{i=0}^{d} f_{t,i}(M_{t-i})$$
$$= \sum_{i=0}^{d}\sum_{t=1}^{T-i} f_{t+i,i}(M_t) = \sum_{t=1}^{T}\sum_{i=0}^{d} f_{t+i,i}(M_t) = \sum_{t=1}^{T} J_t(M_t)$$

Where the first equality holds by separability, the second by shifting the sum index and using the convention $f_{t<d}(\cdot) \doteq 0$ w.l.o.g,[12] and the third by $f_{t>T}(\cdot) \doteq 0$. □

[11] This would result in a sum of the form $\sum_t 1/t \leq \log(T)$.

[12] The adversarial rounds can always be prefixed with zero cost rounds. Alternatively, redefine regret to start from $t = d$ as in see [179, Sec 2.2].

Now we are ready to prove Theorem 5.7:

Proof. Denote with $\pi^\star$ the cost-minimizing policy, and let $M^\star$ be its parametrization. Then, from (5.2):

$$\begin{aligned}\mathcal{R}_T &= \sum_{t=1}^{T} c_t(\boldsymbol{x}_t, \boldsymbol{u}_t) - c_t(\boldsymbol{x}_t(\pi^\star), \boldsymbol{u}_t(\pi^\star)) \\ &\overset{(a)}{\leq} \sum_{t=1}^{T} \Big(\sum_{i=0}^{d} f_t^{(i)}(M_{t-i}) - \sum_{i=0}^{d} f_t^{(i)}(M^\star) \Big) + 2\alpha\epsilon \\ &\overset{(b)}{=} \sum_{t=1}^{T} f_t(M_{t-d}, \dots, M_t) - f_t(M^\star, \dots, M^\star) + 2\alpha\epsilon \\ &\overset{(c)}{=} \underbrace{\sum_{t=1}^{T} F_t(M_t) - F_t(M^\star)}_{\doteq \mathcal{R}_T^F} + 2\alpha\epsilon,\end{aligned}$$

where (a) follows by Lemma 5.8, which gives both an upper bound on the learner cost and lower bound on the benchmark's cost, (b) by writing the sum of $f_t^{(i)}(\cdot)$ as a single function with memory, and (c) by Lemma 5.9, with $f_{t,i}(M) = f_t^{(i)}(M)$, and hence, $J_t(M) = F_t(M)$. To bound the delayed feedback regret $\mathcal{R}_T^F$, we use [86, Thm. 11], which we restate below using the notation of this paper:[13]

Theorem 5.10 ([86, Thm. 11])**.** *Let λ_t be non-decreasing on t defined as in* (5.21)*. Let $r_t(\cdot)$ be λ_t strongly convex regularizer. Then, $\{M_t\}_t$ computed via* (5.20) *ensures that:*

$$\mathcal{R}_T^F \leq 8\kappa_M \max_{t\in[T]} \Delta_{t-d:t-1} + 2\sqrt{5}\kappa_M \sqrt{\sum_{t=1}^{T} \Delta_t^2}. \tag{5.25}$$

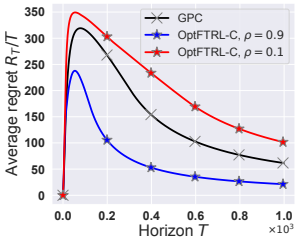
To write Δ_t explicitly, note that $G_{t-d:t}$ can be written as:

$$G_{t-d:t} = \sum_{i=0}^{d} \sum_{j=0}^{d} G_{t-d+i+j}^{(j)}$$

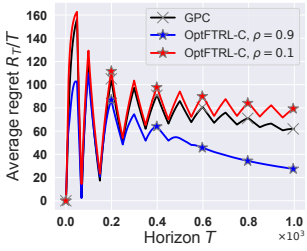
hence we get that

$$\Delta_t = \|G_{t-d:t} - H_t\| \overset{(5.19)}{\leq} \sum_{i=0}^{d-1} \sum_{j=d-i}^{d} \|G_{t-d+i+j}^{(j)} - \tilde{G}_{t-d+i+j}^{(j)}\| + \sum_{j=0}^{d} \Delta_{t+j}^{(j)}$$

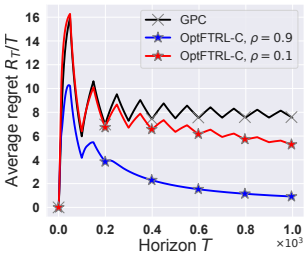
[13] Mapping their notation to ours, we get $\boldsymbol{g}_{t-D:t} = G_{t-d:t}$, $\boldsymbol{h}_t = H_t$, $\boldsymbol{g}_{t-D:t} - h_t = \Delta_t$, $\boldsymbol{a}_{t,F} = 2\kappa_M\Delta_t$, $\boldsymbol{b}_{t,F} = 1/2\Delta_t^2$, and $\alpha = \kappa_M^2$.



(a) Stationary costs.



(b) Alternating costs.



(c) Alternat. costs, low magnitude.

Figure 5.3: The average regret against the optimal policy under various scenarios (cost and disturbances trajectories).

$$= \sum_{i=0}^{d} \sum_{j=d-i}^{d} \Delta_{t-d+i+j}^{(j)} = \sum_{i=0}^{d} \sum_{j=0}^{i} \Delta_{t+j}^{(j+d-i)} = \sum_{i=0}^{d} \sum_{j=i}^{d} \Delta_{t+i}^{(d-j+i)}$$

Substituting back in (5.25) gives the result. □

5

5.5.4. Numerical Example

Recall that `OptFTRL-C` was designed to take advantage of a prediction oracle that forecasts future cost functions with unknown accuracy. Theorem 5.7 then demonstrated that the average policy regret $\mathcal{R}_T/T$ of `OptFTRL-C` always converges to 0, but does so faster for accurate predictions. We therefore plot the policy regret of `OptFTRL-C` when provided with either accurate or inaccurate predictions. The implementation code of the policies `OptFTRL-C`, GPC, and the benchmark $\pi^\star$, as well as the code to reproduce the experiments, is available at the repository [217].

We consider a system with $\boldsymbol{x}, \boldsymbol{u} \in \mathbb{R}^2$, $p = d = 10$,[14] and hence $M \in \mathbb{R}^{2\times 20}$. The dynamics are $A = 0.9 \times \boldsymbol{I}_2, B = \boldsymbol{I}_2$, with perturbation $\boldsymbol{w}_t \in [-1, 1]^2$ of maximum magnitude of $w = \sqrt{2}$. We consider a linear cost $c_t = \langle \boldsymbol{\alpha}_t, \boldsymbol{x}_t \rangle$, with $\boldsymbol{\alpha}_t \in [-1, 1]^2$ and hence $\alpha = \sqrt{2}$. With these choices, we have the upper bound on the gradient $\|G_t\| \leq {}^{\alpha\kappa_B p w}/_{0.1} \leq 300$.

In the accurate prediction case, we set $\tilde{c}_t(\cdot, \cdot) = c_t(\cdot, \cdot)$ with probability $\rho = 0.9$. Otherwise, we set $\tilde{c}_t(\cdot, \cdot)$ to be uniformally random ($\boldsymbol{\alpha}_t \in [-1, 1]^2$). Hence, ρ represents the probability of correctly predicting $c_t(\cdot, \cdot)$, and we sample it at every slot. For inaccurate prediction, we set $\rho = 0.1$. We compare with GPC, which was shown to outperform the classical $\mathcal{H}_2$ and $\mathcal{H}_\infty$ controllers in a wide range of situations.

In scenario (a), The cost trajectory is set as $\boldsymbol{\alpha}_t = (1, 1), \boldsymbol{w}_t = (1, 1), \forall t$. This represents a simple case where the cost function does not fluctuate. It can be seen from Fig. 5.3a that `OptFTRL-C` provides the expected acceleration when the prediction is accurate, achieving an average of 60.2% smaller $\mathcal{R}_T/T$ value compared

[14] The DAC parameter p and the cost's memory d are denoted h and H in [178], where they are also set equal. While both are commonly referred to as "memory", we refer to d also as the delay due to the duality we presented.

	GPC	`OptFTRL-C` $\rho = 0.9$	`OptFTRL-C` $\rho = 0.1$	Optimal
Scenario (a)	314, 694	355, 061	274, 965	376, 198
Scenario (b)	34, 994	69, 468	16, 934	96, 869
Scenario (c)	642	7, 276	2, 928	8, 191

Table 5.1: Accumulated reward (negative cost) $-\sum_{t=1}^{T} c_t(\boldsymbol{x}_t, \boldsymbol{u}_t)$ of the different policies.

to GPC. At the same time, the average regret still *attenuates* at the same $\mathcal{O}(1/\sqrt{T})$ rate even in the case of inaccurate predictions, but with an average performance degradation of 47.3%.

In scenario (b), we deploy *an alternating* cost function. Namely, $\boldsymbol{\alpha}_t$ alternates between $(1, 1)$ and $(-0.5, -0.5)$ every 50 steps. The disturbances are still $\boldsymbol{w}_t = (1, 1), \forall t$. This alternating cost represents an adversarial fluctuation in the cost trajectory, and it is where the non-stochastic framework demonstrates its efficacy. Namely, this fluctuation is enough to violate the guarantees of $\mathcal{H}_2$ controllers, but at the same time, it is small in magnitude, rendering $\mathcal{H}_\infty$ controller overly pessimistic. It is worth noting that the fluctuation observed in Fig. b is attributed to the update rule of GPC. This update directly modifies the decision variables M_{t+1} based on the observed cost. In contrast, FTRL aggregates both past and future costs to determine M_{t+1}. With accurate predictions, this method does not induce much fluctuation as it foresees the upcoming small disturbance. Overall, `OptFTRL-C` with good prediction achieves an improvement of 32% in $\mathcal{R}_T/T$ value over GPC, while having a 13.4% degradation when fed with the inaccurate oracle.

In scenario (c) we also deploy an alternating cost function but with different lower magnitudes. Namely, $\boldsymbol{\alpha}_t$ alternates between $(0.1, 0.1)$ and $(-0.5, -0.5)$ every 50 steps, and $w_t = (0.1, 0.1)$. The goal of this scenario is to show that `OptFTRL-C` can have an advantage over GPC even regardless of the prediction quality through *adapatablity* to "easy" environments (i.e., environments with small gradients). In general, GPC performance takes a hit since its learning rate is tuned with the upper bound for the gradient (i.e., $\alpha = w = \sqrt{2}, T = 10^3$). The alternating frequency, on the other hand, contributes to distinguishing more the effect of good predictions. In this scenario `OptFTRL-C` achieves an improvement of 16.8% and 69.8% over GPC for $\rho = 0.1$ and $\rho = 0.9$, respectively. In summary, `OptFTRL-C` leverages predictions without sacrificing its resilience to their inaccuracy, or to the costs' adversity.

5.6. Conclusion

This chapter examined the Non-Stochastic Control (NSC) problem, a variant of the online convex optimization problem with memory, with the objective of obtaining improved policy regret guarantees. In Part I, we developed algorithms whose policy regret adapts to the complexity of the controlled environment. Leveraging

the Follow-the-Regularized-Leader (FTRL) framework with adaptive regularizers proportional to observed costs, we addressed the unique challenges posed by state-dependent dynamics. Our analysis introduced new techniques for integrating NSC with FTRL, resulting in novel Disturbance-Action Controllers (DAC) that achieve sublinear, data-adaptive policy regret bounds. Part II focused on designing DAC controllers capable of leveraging predictions of unknown quality. The overarching goal was to develop controllers that achieve meaningful policy regret guarantees, with performance enhanced by accurate predictions while remaining robust when predictions fail. By analyzing online learning with memory through the lens of Delayed Online Convex Optimization, we presented the first optimistic DAC controller. While the efficacy of the proposed algorithms has been demonstrated both theoretically and through numerical examples, deploying these techniques in real-world caching systems or other communication network applications remains an avenue for future work.

6

Conclusion

6.1. A Look Back

Returning to the central research question, this thesis set out to investigate methodologies for safely incorporating untrusted predictions into sequential decision-making problems within the context of communication networks, particularly focusing on caching scenarios. Traditional online learning methods excel at providing guarantees under worst-case adversarial conditions, making them suitable for volatile environments typical of modern networks. However, their inherent conservatism prevents them from fully exploiting potentially informative, though imperfect, predictions about the future. On the other hand, modern offline-trained machine learning models demonstrate excellent performance under stable, data-rich scenarios, but their effectiveness significantly degrades in dynamic or unpredictable conditions.

To bridge this gap and address the research question, we identified optimistic learning as a principled framework capable of balancing prediction-adaptivity (i.e., leveraging accurate predictions) with robustness to maintain strong performance guarantees even when predictions fail. In other words, optimistic learning provides a best-of-both-worlds approach: it safely exploits predictions without sacrificing the core robustness of traditional online methods. Optimistic algorithms accomplish this by explicitly incorporating prediction quality into their decision-making logic in a carefully designed manner.

Throughout the thesis, we developed multiple optimistic learning algorithms tailored to diverse network optimization challenges, ranging from continuous resource allocation problems (such as coded caching) to inherently discrete and combinatorial scenarios (like whole-file caching and knapsack-type decisions). Furthermore, we extended the optimistic learning paradigm to handle more stringent metrics and environments, including dynamic regret metric, where optimal decisions evolve over time, and stateful environments where costs are memory-dependent. In each case, we rigorously established formal performance guarantees, ensuring that our proposed methods not only match but often surpass existing approaches. Comple-

menting our theoretical results, we conducted extensive numerical experiments that illustrated the practical relevance and adaptability of these algorithms.

This thesis thus represents the first systematic and comprehensive effort to integrate and extend optimistic learning within the broader context of network optimization. Our contributions include applying and refining existing optimistic algorithms specifically to address unique challenges arising in communication networks, as well as designing novel optimistic learning techniques whose scope and impact extend well beyond networking applications. Collectively, these advancements expand the utility of optimistic learning, providing a robust foundation for future research and enabling wider implementation across diverse domains within this evolving field.

Next, we summarize the key per-chapter contributions of this work.

6.2. Summary of Contributions

In this section, we recapitulate the major contributions of this thesis.

Chapter 2. Optimism for caching: improved guarantees and meta-learning extensions. We applied optimistic learning to bipartite caching networks, demonstrating its effectiveness in improving caching performance under uncertainty. Building upon the seminal Optimistic Follow The Regularized Leader (OFTRL) algorithm, we refined its theoretical guarantees and developed a caching system that adapts to predictions while maintaining robust performance guarantees. Additionally, we introduced an optimistic meta-learning framework that simultaneously identifies the most reliable predictor (when multiple prediction sources are available) and optimizes the caching strategy accordingly. Numerical evaluations confirmed that our proposed methods outperform existing caching benchmarks, underscoring the practicality of optimism in caching applications. Yet, the presented algorithms assume continuous decisions and are thus only applicable for coded-caching.

Chapter 3. Optimism for discrete caching and combinatorial allocation. We extended optimistic learning to discrete caching problems, where storage decisions are inherently combinatorial. Whole-file caching was formulated as a k-set selection problem and, when considering files of varying sizes, as an instance of the knapsack problem. To address this setting, we developed the first discrete optimistic online learning algorithms with provable regret guarantees. In addition to adapting OFTRL for discrete settings, we introduced Optimistic Follow The Perturbed Leader (OFTPL)—an extension of FTPL that improves computational efficiency compared to OFTRL while incurring a slight trade-off in theoretical guarantees. Our results and comprehensive experiments demonstrated that optimism is also a powerful tool in discrete resource allocation, further extending its applicability beyond continuous settings.

Chapter 4. Optimism under a more stringent metric. Moving beyond static regret, the main metric of the previous two chapters, we examined the more challenging dynamic regret metric, where the benchmark evolves over time. Unlike previous chapters, which assume a fixed optimal decision, this setting required tracking a moving comparator. We provided the first analysis of OFTRL under dynamic comparators, revealing that optimism plays an even greater role in adapting to shifting benchmarks. A key contribution of this work was our pruning-based refinement

of OFTRL, which yields tighter dynamic regret bounds that scale with prediction accuracy. Through theoretical analysis and numerical validation, we demonstrated that optimistic learning effectively balances prediction adaptivity and robustness in settings where the optimal solution is non-stationary.

Chapter 5. Optimism & functions with memory. Finally, we extended the optimistic learning framework to stateful systems, where past decisions influence future costs. This generalization allows cost functions to depend on a history of prior actions, introducing additional complexity in the learning process. We focused on the Non-Stochastic Control (NSC) paradigm and identified two key limitations in existing approaches: the lack of data-dependent adaptivity and the lack of prediction adaptivity. To address these limitations, we introduced the first optimistic learning algorithms designed explicitly for functions with memory, demonstrating that horizon-based predictions, which extend beyond single-step forecasts, are essential for optimizing performance in this setting. Furthermore, we established a fundamental connection between memory, delay, and optimism, offering new insights into the interplay between these three key variants of online learning.

In summary, this thesis provides a systematic and rigorous exploration of optimistic learning in networked systems. By integrating predictions into online learning algorithms while maintaining worst-case guarantees, our work advances state-of-the-art in caching, network optimization, and sequential decision-making. Ultimately, this research contributes toward more intelligent and adaptive network management solutions, bridging the gap between theoretical learning frameworks and practical deployment in dynamic environments.

6.3. Future directions

6.3.1. Integrating Distance-based & Directional Optimism

As highlighted in the introduction of several chapters, an alternative approach in optimistic learning is to assess the effectiveness of predictions using their directional alignment with the actual costs rather than their norm-based distance. This error measurement method leads to regret bounds of a different nature ranging from $\log T$ to $\sqrt{T}$, [172, 218], which suggests that predictions correlated with actual costs can be leveraged for expediting learning, even if they are not point-wise accurate (the norm-based error is large). This, in turn, allows us to incorporate a broader set of prediction mechanisms and therefore extend the set of network management problems that can benefit from this toolbox.

In this context, an important research direction is to investigate how correlation-based optimism can be seamlessly integrated with the norm-based approach presented in this thesis. The primary challenge in doing so lies in the structural assumptions imposed by correlation-based methods; most notably, the technical requirement for strong convexity in the decision set. Future work could explore ways to relax these convexity requirements to eventually develop hybrid approaches that switch dynamically between norm-based and directional-based optimism depending on the problem characteristics. The ideal outcome here would be to obtain bounds that depend on the minimum error among these two approaches.

Besides integration, a pertinent future direction is the theoretical refinement and empirical evaluation of correlation-based optimism in real-world network applications. So far, research in this area has been limited, with only few exceptions. The work in [219] explored directional optimism for communication, applying it to opportunistic channel selection and mobile crowd sensing. Similarly, [220] investigated the same applications but in a decentralized setting, where predictions take the form of messages from potentially malicious neighbors, making them inherently unreliable. Such works show the potential of this idea and can be further explored.

6.3.2. Unifying Adversarial & Stochastic Environments

Another future direction for optimistic learning in the context of communication networks is to leverage the stochastically extended adversarial (SEA) model [130], which interpolates between stochastic and adversarial environments. This model essentially assumes the environment is not entirely adversarial but instead exhibits some stochastic structure, albeit with occasional distributional shifts. This condition allows for designing algorithms that leverage stochastic regularity while remaining robust to adversarial variations. It also subsumes the fully adversarial environments (addressed in this tutorial) and the fully stochastic settings (stochastic optimization), and the in-between spectrum. Recent works such as [130, 163, 221], have explored regret bounds that depend on both the stochastic variance and the adversarial variation of the gradients. These results indicate that in a predominantly stochastic environment with occasional adversarial perturbations, it is possible to improve the regret bounds compared to those achieved in fully adversarial settings. This is very useful for those communication networks that operate under benign conditions most of the time (stochastic setting), with occasional disruptions due to, e.g., an attack, that make the conditions adversarial for only a specific time window. To the best of our knowledge, the SEA model has not yet been applied to communication problems, leaving it a promising future step.

Moreover, in this context, it is intriguing to investigate how predictions should be designed or, put differently, what type of predictions are beneficial. Unlike typical optimism, where predictions focus on the next gradient (or function), in SEA, the learner could use forecasts about the underlying distributional structure, its parameters, the timing of distribution shifts, or the presence of stochastic elements.

6.3.3. Reductions among Metrics

Other important learning metrics, such as the competitive ratio and adaptive regret offer valuable perspectives worth exploring in future research. Some studies have explored the connection between the competitive ratio and various regret-based metrics. For instance, [74] examined the relationship between competitive ratio and static regret, while [222] investigated online learning approaches that simultaneously address competitive ratio and dynamic regret with switching costs, illustrating the potential for algorithms to balance (or hedge on) multiple learning criteria. Furthermore, the relationship between competitive ratio and policy regret was studied in [196]; and several works examined the interplay among different regret notions per se. For example, [223] proposed algorithms that simultaneously achieve guar-

antees in both adaptive and dynamic regret; and [173] provided insights into the relationship between static and dynamic regret through a generalized "path-length" complexity measure. Future research could delve into the relationships among these learning metrics and potentially devise further reductions among them, particularly through the lens of optimistic learning (i.e., considering untrusted predictions). Understanding these intricate relationships can ultimately guide the design of universal, optimistic algorithms that guarantee effective learning with respect to multiple criteria simultaneously, which is essential since different learning metrics are suitable for different network problems.

6.3.4. Systems with States and Memory

In Chapter 5, we explored optimistic learning for stateful problems where the decisions influence an evolving system state and, through that, also the cost function, and we focused on linear time-invariant dynamical systems. However, this important first step captures only a subset of broader stateful decision-making problems. Therefore, it remains an open question whether it is possible, and how, to extend optimistic learning to a wider class of stateful problems, e.g., when the system dynamics vary with time or based on a non-linear rule, or when the system behavior is governed by an underlying Markov Decision Process (MDPs). Indeed, MDPs and the associated Reinforcement Learning (RL) algorithms have an extremely wide application range in communication systems, and in that regard, any such extension of optimistic learning will be impactful. For an introduction to MDPs in the context of wireless communication networks, see the survey in [224], and for an overview of modern reinforcement learning techniques in networks, see [225]. Interestingly, no-regret algorithms such as FTRL [197] and OFTPL [226] have already been investigated in the context of MDPs, highlighting the promising potential for extending their optimistic variants to this important class of problems.

6.3.5. Constant-aware Bounds

It is crucial to develop the theoretical foundations of optimistic learning further to ensure that regret bounds do not degrade unfavorably when predictions are highly inaccurate. While this robustness is already ensured in some settings, in others, the introduction of optimism can come at the cost of worse constants in the bounds. Now, in theoretical OCO studies and even in ML applications, the focus has been predominantly on the convergence rate of the regret bounds, i.e., on the dependency of the regret on the time horizon T. Nevertheless, in communication problems, we are often interested in the dependency of the regret on other system parameters. A notable example is caching, where a regret bound that increases with the number of files is highly undesirable. A step in this direction is the work in [227], which replaces the typical quadratic dependence on the prediction error with a Huber-style loss. The distinction becomes relevant when prediction errors are large: both approaches yield sublinear regret in T, but the Huber loss leads to significantly milder degradation, with constants scaling only with the square root of those in the standard case. Further exploration of such techniques may help ensure that optimistic learning remains competitive even in poorly predicted network environments.

3A

Appendix of Chapter 3

3A.1. Madow's Sampling Algorithm

Algorithm 13 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 [136] for futher details.

Algorithm 13: Madow's Sampling (`MadowSample`)

Input: $\hat{x} \in [0,1]^N, \sum x_{i \in \mathcal{N}} \leq C$.
Output: Random set S, s.t $|S| = C$ and $\Pr(i \in S) = x_i$
Sample a uniformly random scalar $U \in [0,1]$
Define the cumulative probabilities $\pi_0 = 0, \quad \pi_i = \pi_{i-1} + \hat{x}_i, \quad \forall\ 1 \leq i \leq N$
for $i = 0, 1, \ldots, C$ **do**
  $S \leftarrow S \cup \{j : \pi_{j-1} \leq U + i < \pi_j\}$
end
return S

3A.2. 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 3.6 are proved in [127, Lem. 2.1].

Algorithm 14: Dependent Rounding (DepRound)

1 **Input**: $a \in [0,1]^N, s \in R_+^N$.
2 **Output**: b satisfying points in lemma-3.6.
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** $0 \leq s_i\ a_i + s_j\ a_j \leq \min\{a_i, a_j\}$ **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** $a_i \leq s_i\ a_i + s_j\ a_j \leq a_j$ **then**
Set $a_i = 1$ with probability a_i. With the remaining probability set $a_i = 0$
end
7 **if** $a_j \leq s_i\ a_i + s_j\ a_j \leq a_i$ **then**
Set $a_j = 1$ with probability a_j. With the remaining probability set $a_j = 0$
end
8 **if** $\max\{a_i, a_j\} \leq s_i\ a_i + s_j\ a_j \leq a_i + a_j$ **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

3A

4A

Appendix of Chapter 4

4A.1. OptFPRL

Below, we restate the main ingredients of `OptFPRL` using a more detailed presentation.

4A.1.1. Update Rule

Recall the notation $\boldsymbol{a}_{1:t} \doteq \sum_{\tau=1}^{t} \boldsymbol{a}_\tau$. The update rule of `OptFPRL` is:

$$\boldsymbol{x}_{t+1} = \arg\min_{\boldsymbol{x}} \big\{ \langle \underbrace{\boldsymbol{p}_{1:t}}_{\text{State vector}}, \boldsymbol{x} \rangle + \underbrace{r_{1:t}(\boldsymbol{x})}_{\text{Incremental regularizers}} + \underbrace{\tilde{f}_{t+1}(\boldsymbol{x})}_{\text{Prediction}} + \underbrace{I_{\mathcal{X}}(\boldsymbol{x})}_{\text{Set constraint}} \big\}. \tag{4A.1}$$

where each $\boldsymbol{p}_t$ is the sum of the linearization of the cost function $f_t(\boldsymbol{x}_t)$, and some choice from the cone $\mathcal{N}_{\mathcal{X}}(\boldsymbol{x}_t)$.

$$\boldsymbol{p}_t = \boldsymbol{g}_t + \boldsymbol{g}_t^I, \quad \boldsymbol{g}_t \in \partial f_t(\boldsymbol{x}_t), \quad \boldsymbol{g}_t^I \in \partial I_{\mathcal{X}}(\boldsymbol{x}_t) = \mathcal{N}_{\mathcal{X}}(\boldsymbol{x}_t). \tag{4A.2}$$

4A.1.2. Regularizer

We use the scaled Euclidean regularizer,

$$r_t(\boldsymbol{x}) = \frac{\sigma_t}{2}\|\boldsymbol{x}\|^2,$$

where σ_t is the strong convexity parameter that will be set in every version of the algorithm. Note that the sum $r_{1:t}(\cdot)$ is $\sigma_{1:t}$-strongly convex with respect to $\|\cdot\|$, or equivalently 1-strongly convex with respect to $\|\cdot\|_t \doteq \sqrt{\sigma_{1:t}}\|\cdot\|$.

4A.1.3. Pruning

Here, we present a mechanism for selecting the vectors $\boldsymbol{g}_t^I$ from the cone $\mathcal{N}_{\mathcal{X}}(\boldsymbol{x}_t)$. Define the unconstrained iterate $\boldsymbol{x}_t^{\text{uc}}$, which is obtained by solving the update rule

without the indicator function

$$\boldsymbol{x}_{t+1}^{\mathrm{uc}} = \arg\min_{\boldsymbol{x}} \langle \boldsymbol{p}_{1:t}, \boldsymbol{x} \rangle + r_{1:t}(\boldsymbol{x}) + \tilde{f}_{t+1}(\boldsymbol{x}).$$

Pruning will depend on whether this iterate belongs to the set $\mathcal{X}$ or not.

The unconstrained iterate $\boldsymbol{x}_t^{\mathrm{uc}}$ always exists and is unique due to the strong convexity of $r_{1:t}(\cdot)$. However, its membership in $\mathcal{X}$ depends on the exact degree of strong convexity. When $\sigma_{1:t} = 0$, we consider $\boldsymbol{x}_t^{\mathrm{uc}} \notin \mathcal{X}$, as the minimizer of an unconstrained linear function does not exist.

We select $\boldsymbol{g}_t^I$ as follows: for $t = 1$, set $\boldsymbol{g}_1^I = -\boldsymbol{g}_1$ if $\epsilon_1 = \sigma_1 = 0$, and $\boldsymbol{g}_1^I = 0$ otherwise. For all $t \geq 2$:

$$\boldsymbol{g}_t^I = \begin{cases} -(\boldsymbol{p}_{1:t-1} + \tilde{\boldsymbol{g}}_t + \sigma_{1:t-1}\boldsymbol{x}_t) & \text{if } \boldsymbol{x}_t^{\mathrm{uc}} \notin \mathcal{X} \\ 0 & \text{otherwise.} \end{cases} \tag{4A.3}$$

Recall that this is a valid choice because when $\boldsymbol{x}_t^{\mathrm{uc}} \notin \mathcal{X}$, then $\boldsymbol{x}_t \in \mathbf{bd}(\mathcal{X})$. Hence, $\mathcal{N}_{\mathcal{X}}(\boldsymbol{x}_t)$ contains elements other than 0, and in particular $-(\boldsymbol{p}_{1:t-1} + \tilde{\boldsymbol{g}}_t + \sigma_{1:t-1}\boldsymbol{x}_t)$. This is a direct result of the optimality condition for the constrained iterate (4A.1), see, e.g., [113, Thm. 3.67]. For the second case, 0 is always a valid subgradient of the indicator function since in this case $\boldsymbol{x}_t^{\mathrm{uc}} = \boldsymbol{x}_t \in \mathcal{X}$.

4A.1.4. Dual-perspective

To provide a dual view of the proposed FTRL variant, we look at the update step through the lens of dual maps.

First, recall the definition of the convex conjugate $r^\star(\cdot)$ of a closed convex function $r(\cdot)$:

$$r^\star(\boldsymbol{y}) \doteq \sup_{\boldsymbol{x}} \langle \boldsymbol{x}, \boldsymbol{y} \rangle - r(\boldsymbol{x}).$$

The update of FTRL can be expressed using the above definition applied to a potentially time-varying $r(\cdot)$. Namely, the standard FTRL update can be expressed as:

$$\boldsymbol{x}_{t+1}^{\mathrm{RL}} = \arg\min_{x} \langle \boldsymbol{g}_{1:t}, \boldsymbol{x} \rangle + r_{0:t}(\boldsymbol{x}) = \nabla r_{0:t}^\star(-\boldsymbol{g}_{1:t}),$$

where $r_{0:t}^\star(\cdot)$ is the conjugate of the cumulative regularizer $r_{1:t}(\cdot)$, restricted to $\mathcal{X}$ via $r_0 = I_{\mathcal{X}}$. From this viewpoint, FTRL maintains the state as cumulative gradients in dual space.

The update of `OptFPRL` can be interpreted as:

$$\boldsymbol{x}_{t+1} = \arg\min_{x} \langle \boldsymbol{p}_{1:t}, \boldsymbol{x} \rangle + r_{0:t}(\boldsymbol{x}) = \nabla r_{0:t}^\star(-\boldsymbol{p}_{1:t}) = \nabla r_{0:t}^\star(\nabla r_{1:t-k-1}(\boldsymbol{x}_{t-k}) - \boldsymbol{g}_{t-k:t}),$$

where $t-k$ denotes the most recent step at which we chose to prune. The last equality holds by the definition of $\boldsymbol{g}_k^I$ (assuming no predictions). Intuitively, we retain explicit gradient history only since the last pruning step $t-k$, while summarizing earlier history implicitly via the dual mapping of $\boldsymbol{x}_{t-k}$. The crux of the paper is showing that the way history is split (explicitly tracked after pruning, and implicitly captured before) is what controls dynamic regret.

4A.1.5. Regret Characterization

Define the dynamic regret metric against any set of comparators $\{\boldsymbol{u}_t\}_t$ as:

$$\mathcal{R}_T \doteq \sum_{t=1}^{T} f_t(\boldsymbol{x}_t) - f_t(\boldsymbol{u}_t) = \sum_{t=1}^{T} j_t(\boldsymbol{x}_t) - j_t(\boldsymbol{u}_t) \leq \sum_{t=1}^{T} \langle \boldsymbol{p}_t, \boldsymbol{x}_t - \boldsymbol{u}_t \rangle, \quad \text{(4A.4)}$$

where $j_t(\boldsymbol{x}) \doteq f_t(\boldsymbol{x}) + I_{\mathcal{X}}(\boldsymbol{x})$. The inequality holds by the linearization principle of convex functions [156, Sec. 2.4], noticing that we indeed have $\boldsymbol{p}_t \in \partial j_t(\boldsymbol{x}_t)$ due the definition of $\boldsymbol{p}_t$ in (4A.2), and the fact that the subdifferential of the sum contains the sum of the subdifferentials (e.g., [28, Thm. 22]).

4A.2. Missing Proofs for Section 4.5

4A.2.1. The Strong Dynamic Optimistic FTRL Lemma

Lemma 4A.1. *(Strong Dynamic Optimistic FTRL). Let $\{f_t(\cdot), \tilde{f}_t(\cdot), \boldsymbol{u}_t\}_{t=1}^T$ be an arbitrary set of functions, predicted functions, and comparators within $\mathcal{X}$, respectively. Let $r_t(\cdot)$ be non-negative regularization functions such that*

$$\boldsymbol{x}_{t+1} \doteq \arg\min_{\boldsymbol{x}} h_{0:t}(\boldsymbol{x}) + \tilde{f}_{t+1}(\boldsymbol{x})$$

is well-defined, where $h_0(\boldsymbol{x}) \doteq I_{\mathcal{X}}(\boldsymbol{x})$, and $\forall t \geq 1$:

$$h_t(\boldsymbol{x}) \doteq \langle \boldsymbol{p}_t, \boldsymbol{x} \rangle + r_t(\boldsymbol{x}), \quad \boldsymbol{p}_t \in \partial j_t(\boldsymbol{x}_t).$$

Then, the algorithm that selects the actions $\boldsymbol{x}_{t+1}, \forall t$ achieves the following dynamic regret bound:

$$\mathcal{R}_T \leq \sum_{t=1}^{T} \overbrace{h_{0:t}(\boldsymbol{x}_t) - h_{0:t}(\boldsymbol{x}_{t+1}) - r_t(\boldsymbol{x}_t)}^{(I)} + \sum_{t=1}^{T-1} \underbrace{h_{0:t}(\boldsymbol{u}_{t+1}) - h_{0:t}(\boldsymbol{u}_t)}_{(II)} + r_t(\boldsymbol{u}_t).$$

Proof.

$$\begin{aligned}
&\sum_{t=1}^{T} h_t(\boldsymbol{x}_t) - \sum_{t=1}^{T} h_t(\boldsymbol{u}_t) - \tilde{f}_{T+1}(\boldsymbol{u}_T) \\
&\quad = \sum_{t=1}^{T} \left(h_{0:t}(\boldsymbol{x}_t) - h_{0:t-1}(\boldsymbol{x}_t)\right) - \left(\sum_{t=1}^{T} \left(h_{0:t}(\boldsymbol{u}_t) - h_{0:t-1}(\boldsymbol{u}_t)\right) + \tilde{f}_{T+1}(\boldsymbol{u}_T)\right) \\
&\quad = \sum_{t=1}^{T} h_{0:t}(\boldsymbol{x}_t) - \sum_{t=1}^{T} h_{0:t-1}(\boldsymbol{x}_t) - \left(\sum_{t=1}^{T} h_{0:t}(\boldsymbol{u}_t) + \tilde{f}_{T+1}(\boldsymbol{u}_T) - \sum_{t=1}^{T} h_{0:t-1}(\boldsymbol{u}_t)\right) \\
&\quad = \sum_{t=1}^{T} h_{0:t}(\boldsymbol{x}_t) - \sum_{t=0}^{T-1} h_{0:t}(\boldsymbol{x}_{t+1}) - \left(h_{0:T}(\boldsymbol{u}_T) + \tilde{f}_{T+1}(\boldsymbol{u}_T) + \right.
\end{aligned}$$

$$
\begin{aligned}
&\sum_{t=1}^{T-1} h_{0:t}(\boldsymbol{u}_t) - \sum_{t=0}^{T-1} h_{0:t}(\boldsymbol{u}_{t+1})\big) \\
&\leq \sum_{t=1}^{T} h_{0:t}(\boldsymbol{x}_t) - \sum_{t=1}^{T-1} h_{0:t}(\boldsymbol{x}_{t+1}) - \big(h_{0:T}(\boldsymbol{x}_{T+1}) + \tilde{f}_{T+1}(\boldsymbol{x}_{T+1}) \\
&\quad + \sum_{t=1}^{T-1} h_{0:t}(\boldsymbol{u}_t) - \sum_{t=1}^{T-1} h_{0:t}(\boldsymbol{u}_{t+1})\big) \\
&= \sum_{t=1}^{T} h_{0:t}(\boldsymbol{x}_t) - \sum_{t=1}^{T} h_{0:t}(\boldsymbol{x}_{t+1}) - \left(\sum_{t=1}^{T-1} h_{0:t}(\boldsymbol{u}_t) - \sum_{t=1}^{T-1} h_{0:t}(\boldsymbol{u}_{t+1}))\right) - \tilde{f}_{T+1}(\boldsymbol{x}_{T+1}).
\end{aligned}
$$

The inequality holds because of the update rule for each $\boldsymbol{x}_{t+1}$ for any t. I.e.,

$$h_{0:T}(\boldsymbol{x}_{T+1}) + \tilde{f}_{T+1}(\boldsymbol{x}_{T+1}) \leq h_{0:T}(\boldsymbol{u}_T) + \tilde{f}_{T+1}(\boldsymbol{u}_T).$$

Also, $h_0(\boldsymbol{u}_{t+1}) = 0, \forall t \leq T-1$.

Writing h_t of the LHS explicitly we get

$$
\begin{aligned}
&\sum_{t=1}^{T} \left(\langle \boldsymbol{p}_t, \boldsymbol{x}_t\rangle + r_t(\boldsymbol{x}_t) - \langle \boldsymbol{p}_t, \boldsymbol{u}_t\rangle - r_t(\boldsymbol{u}_t)\right) - \tilde{f}_{T+1}(\boldsymbol{u}_T) \\
&\qquad \leq \sum_{t=1}^{T} \left(h_{0:t}(\boldsymbol{x}_t) - h_{0:t}(\boldsymbol{x}_{t+1})\right) + \sum_{t=1}^{T-1} \left(h_{0:t}(\boldsymbol{u}_{t+1}) - h_{0:t}(\boldsymbol{u}_t)\right) - \tilde{f}_{T+1}(\boldsymbol{x}_{T+1}).
\end{aligned}
$$

Since $\tilde{f}_{T+1}(\cdot)$ does not affect the algorithm, we can set it to 0. Rearranging:

$$
\begin{aligned}
&\sum_{t=1}^{T} \langle \boldsymbol{p}_t, \boldsymbol{x}_t\rangle - \langle \boldsymbol{p}_t, \boldsymbol{u}_t\rangle \\
&\leq \sum_{t=1}^{T} \left(h_{0:t}(\boldsymbol{x}_t) - h_{0:t}(\boldsymbol{x}_{t+1}) - r_t(\boldsymbol{x}_t) + r_t(\boldsymbol{u}_t)\right) + \sum_{t=1}^{T-1} \left(h_{0:t}(\boldsymbol{u}_{t+1}) - h_{0:t}(\boldsymbol{u}_t)\right).
\end{aligned}
$$

Noting that by (4A.4), the LHS upper-bounds the regret, we get the result. □

4A.2.2. Bounding the First Part (I):

We bound **(I)** : $h_{0:t}(\boldsymbol{x}_t) - h_{0:t}(\boldsymbol{x}_{t+1}) - r_t(\boldsymbol{x}_t)$ in two ways, which results in the $\min(\cdot,\cdot)$ term. We present in this subsection Lemmas 4A.2 and 4.6, corresponding to each argument of the $\min(\cdot,\cdot)$.

First, we begin with Lemma 4A.1, which provides an additional characterization of the iterate $\boldsymbol{x}_t$ as the minimizer not only of the original update rule, but also of a related linearized expression.

Lemma 4A.1. *For any* $\boldsymbol{x}_t \in \mathcal{X}$, $\boldsymbol{x}_t = \arg\min_{\boldsymbol{x}} h_{0:t-1}(\boldsymbol{x}) + \tilde{f}_t(\boldsymbol{x}) \implies \boldsymbol{x}_t = \arg\min_{\boldsymbol{x}} h_{0:t-1}(\boldsymbol{x}) + \langle \tilde{\boldsymbol{g}}_t, \boldsymbol{x}\rangle + \langle \boldsymbol{g}_t^I, \boldsymbol{x}\rangle$, *where* $\tilde{\boldsymbol{g}}_t \in \partial \tilde{f}_t(\boldsymbol{x}_t)$ *and* $\boldsymbol{g}_t^I$ *is selected according to* (4A.3).

Proof. We are given that $\boldsymbol{x}_t = \arg\min_{\boldsymbol{x}} h_{0:t-1}(\boldsymbol{x}) + \tilde{f}_t(\boldsymbol{x})$. Thus, from the optimality condition for constrained problems (e.g., [113, Thm. 3.67]), the negative (sub)gradient must belong to the normal cone at $\boldsymbol{x}_t$:

$$-\nabla h_{1:t-1}(\boldsymbol{x}_t) - \tilde{\boldsymbol{g}}_t \in \mathcal{N}_{\mathcal{X}}(\boldsymbol{x}_t) \quad \text{(By definition of } \tilde{\boldsymbol{g}}_t\text{)}. \tag{4A.5}$$

Next, we examine the optimality condition for $h_{0:t-1}(\boldsymbol{x}) + \langle \tilde{\boldsymbol{g}}_t, \boldsymbol{x}\rangle + \langle \boldsymbol{g}_t^I, \boldsymbol{x}\rangle$ and show that $\boldsymbol{x}_t$ satisfies it. For $\boldsymbol{y}$ to be minimizer of $h_{0:t-1}(\boldsymbol{x}) + \langle \tilde{\boldsymbol{g}}_t, \boldsymbol{x}\rangle + \langle \boldsymbol{g}_t^I, \boldsymbol{x}\rangle$, it must satisfy:

$$-\nabla h_{1:t-1}(\boldsymbol{y}) - \tilde{\boldsymbol{g}}_t - \boldsymbol{g}_t^I \in \mathcal{N}_{\mathcal{X}}(\boldsymbol{y}).$$

Substituting $\boldsymbol{x}_t$ in the above we get

$$-\nabla h_{1:t-1}(\boldsymbol{x}_t) - \tilde{\boldsymbol{g}}_t - \boldsymbol{g}_t^I \in \mathcal{N}_{\mathcal{X}}(\boldsymbol{x}_t). \tag{4A.6}$$

Now, note that according to the linearization choices in (4A.3), we have that either (i): $\boldsymbol{g}_t^I = 0$, or (ii): $\boldsymbol{g}_t^I = -(\boldsymbol{p}_{1:t-1} + \tilde{\boldsymbol{g}}_t + \sigma_{1:t-1}\boldsymbol{x}_t)$.

In case (i), (4A.6) reduces to the given (4A.5), meaning that $\boldsymbol{x}_t$ satisfies (4A.6).

In case (ii), Note that $-(\boldsymbol{p}_{1:t-1} + \tilde{\boldsymbol{g}}_t + \sigma_{1:t-1}\boldsymbol{x}_t) = -\nabla h_{1:t-1}(\boldsymbol{x}_t) - \tilde{\boldsymbol{g}}_t$, and hence (4A.6) reduces to $0 \in \mathcal{N}_{\mathcal{X}}(\boldsymbol{x}_t)$, which is always true. Thus, $\boldsymbol{x}_t$ satisfies (4A.6) in this case too.

It follows that $\boldsymbol{x}_t$ satisfies the optimality condition for $h_{0:t-1}(\boldsymbol{x}) + \langle \tilde{\boldsymbol{g}}_t, \boldsymbol{x}\rangle + \langle \boldsymbol{g}_t^I, \boldsymbol{x}\rangle$ and hence is a minimizer[1] thereof. □

Lemma 4A.2. *Let each function $h_{0:t}(\cdot)$ be 1-strongly convex with respect to a norm $\|\cdot\|_t$ defined as in Lemma 4.5. Let $\boldsymbol{x}_t = \arg\min_{\boldsymbol{x}} h_{0:t-1}(\boldsymbol{x}) + \tilde{f}_t(\boldsymbol{x})$. Then, we have the inequality*

$$h_{0:t}(\boldsymbol{x}_t) - h_{0:t}(\boldsymbol{x}_{t+1}) - r_t(\boldsymbol{x}_t) \le 2R\epsilon_t.$$

Proof.

$$\begin{aligned}
&h_{0:t}(\boldsymbol{x}_t) - h_{0:t}(\boldsymbol{x}_{t+1}) - r_t(\boldsymbol{x}_t) \\
&= h_{0:t-1}(\boldsymbol{x}_t) + \langle \boldsymbol{p}_t, \boldsymbol{x}_t\rangle - h_{0:t-1}(\boldsymbol{x}_{t+1}) - \langle \boldsymbol{p}_t, \boldsymbol{x}_{t+1}\rangle - r_t(\boldsymbol{x}_{t+1}) \\
&\overset{(a)}{\le} h_{0:t-1}(\boldsymbol{x}_t) + \langle \boldsymbol{p}_t, \boldsymbol{x}_t\rangle - h_{0:t-1}(\boldsymbol{x}_{t+1}) - \langle \boldsymbol{p}_t, \boldsymbol{x}_{t+1}\rangle \\
&\overset{(b)}{\le} h_{0:t-1}(\boldsymbol{x}_t) + \langle \boldsymbol{p}_t, \boldsymbol{x}_t\rangle - h_{0:t-1}(\boldsymbol{y}_t) - \langle \boldsymbol{p}_t, \boldsymbol{y}_t\rangle,
\end{aligned} \tag{4A.7}$$

where (a) follows by dropping the negative $-r_t(\boldsymbol{x}_{t+1})$ term and (b) by defining $\boldsymbol{y}_t \doteq \arg\min_{\boldsymbol{x}} h_{0:t-1}(\boldsymbol{x}) + \langle \boldsymbol{p}_t, \boldsymbol{x}\rangle$. Then,

$$\begin{aligned}
&h_{0:t-1}(\boldsymbol{x}_t) + \langle \boldsymbol{p}_t, \boldsymbol{x}_t\rangle - h_{0:t-1}(\boldsymbol{y}_t) - \langle \boldsymbol{p}_t, \boldsymbol{y}_t\rangle \\
&= h_{0:t-1}(\boldsymbol{x}_t) \ + \langle \tilde{\boldsymbol{g}}_t + \boldsymbol{g}_t^I, \boldsymbol{x}_t\rangle + \langle \boldsymbol{p}_t, \boldsymbol{x}_t\rangle - h_{0:t-1}(\boldsymbol{y}_t) - \langle \boldsymbol{p}_t, \boldsymbol{y}_t\rangle \ - \langle \tilde{\boldsymbol{g}}_t + \boldsymbol{g}_t^I, \boldsymbol{x}_t\rangle
\end{aligned}$$

[1] Since $h_{0:t}(\cdot)$ is strongly convex, the minimizer of the two expressions in the lemma statement always exist and unique. In the case $\sigma_{1:t} = 0$, we abuse notation by writing " $= \arg\min$ " instead of " $\in \arg\min$ ". This is not problematic because we do not require the equivalence of the minimizers.

$$\overset{(c)}{\leq} h_{0:t-1}(\boldsymbol{y}_t) + \langle \tilde{\boldsymbol{g}}_t + \boldsymbol{g}_t^I, \boldsymbol{y}_t \rangle + \langle \boldsymbol{p}_t, \boldsymbol{x}_t \rangle - h_{0:t-1}(\boldsymbol{y}_t) - \langle \boldsymbol{p}_t, \boldsymbol{y}_t \rangle - \langle \tilde{\boldsymbol{g}}_t + \boldsymbol{g}_t^I, \boldsymbol{x}_t \rangle$$
$$= \langle \tilde{\boldsymbol{g}}_t + \boldsymbol{g}_t^I, \boldsymbol{y}_t - \boldsymbol{x}_t \rangle + \langle \boldsymbol{p}_t, \boldsymbol{x}_t - \boldsymbol{y}_t \rangle = \langle \boldsymbol{p}_t, \boldsymbol{x}_t - \boldsymbol{y}_t \rangle - \langle \tilde{\boldsymbol{g}}_t + \boldsymbol{g}_t^I, \boldsymbol{x}_t - \boldsymbol{y}_t \rangle$$
$$= \langle \boldsymbol{p}_t - \tilde{\boldsymbol{g}}_t - \boldsymbol{g}_t^I, \boldsymbol{x}_t - \boldsymbol{y}_t \rangle = \langle \boldsymbol{g}_t - \tilde{\boldsymbol{g}}_t, \boldsymbol{x}_t - \boldsymbol{y}_t \rangle \leq 2R\|\boldsymbol{g}_t - \tilde{\boldsymbol{g}}_t\| = 2R\epsilon_t,$$

where we added & subtracted $\langle \tilde{\boldsymbol{g}}_t + \boldsymbol{g}_t^I, \boldsymbol{x}_t \rangle$ in the first equality, and (c) holds because $\boldsymbol{x}_t$ is the minimizer of $h_{0:t-1}(\boldsymbol{x}) + \langle \tilde{\boldsymbol{g}}_t + \boldsymbol{g}_t^I, \boldsymbol{x} \rangle$ (shown by Lemma 4A.1). Overall, we obtain

$$h_{0:t}(\boldsymbol{x}_t) - h_{0:t}(\boldsymbol{x}_{t+1}) - r_t(\boldsymbol{x}_t) \leq 2R\epsilon_t. \tag{4A.8}$$

□

To produce the other argument in the $\min(\cdot, \cdot)$, we use the following lemma.

Lemma 4A.3. *Let each function $h_{0:t}(\cdot)$ be 1-strongly convex with respect to a norm $\|\cdot\|_t$ defined as in Lemma 4.5. Let $\boldsymbol{x}_t = \arg\min_{\boldsymbol{x}} h_{0:t-1}(\boldsymbol{x}) + \tilde{f}_t(\boldsymbol{x})$. Then, we have the inequality*

$$h_{0:t}(\boldsymbol{x}_t) - h_{0:t}(\boldsymbol{x}_{t+1}) - r_t(\boldsymbol{x}_t) \leq \frac{1}{2}\|\boldsymbol{g}_t - \tilde{\boldsymbol{g}}_t\|_{t-1,*}^2 \ .$$

Proof.

$$\begin{aligned} &h_{0:t}(\boldsymbol{x}_t) - h_{0:t}(\boldsymbol{x}_{t+1}) - r_t(\boldsymbol{x}_t) \\ &\quad = h_{0:t-1}(\boldsymbol{x}_t) + \langle \boldsymbol{p}_t, \boldsymbol{x} \rangle - h_{0:t-1}(\boldsymbol{x}_{t+1}) - \langle \boldsymbol{p}_t, \boldsymbol{x}_{t+1} \rangle - r_{t+1}(\boldsymbol{x}_{t+1}) \\ &\quad \leq h_{0:t-1}(\boldsymbol{x}_t) + \langle \boldsymbol{p}_t, \boldsymbol{x}_t \rangle - h_{0:t-1}(\boldsymbol{x}_{t+1}) - \langle \boldsymbol{p}_t, \boldsymbol{x}_{t+1} \rangle \\ &\quad \leq h_{0:t-1}(\boldsymbol{x}_t) + \langle \boldsymbol{p}_t, \boldsymbol{x}_t \rangle - h_{0:t-1}(\boldsymbol{y}_t) - \langle \boldsymbol{p}_t, \boldsymbol{y}_t \rangle, \quad \boldsymbol{y}_t \doteq \arg\min_{\boldsymbol{x}} h_{0:t-1}(\boldsymbol{x}) + \langle \boldsymbol{p}_t, \boldsymbol{x} \rangle \end{aligned}$$

Where again the inequality follows by dropping the non-positive $r_{t+1}(\cdot)$. Next, we invoke Lemma 4A.7 with

$$\phi_1(\boldsymbol{x}) \doteq h_{0:t-1}(\boldsymbol{x}) + \langle \tilde{\boldsymbol{g}}_t, \boldsymbol{x} \rangle + \langle \boldsymbol{g}_t^I, \boldsymbol{x} \rangle,$$
$$\phi_2(\boldsymbol{x}) \doteq h_{0:t-1}(\boldsymbol{x}) + \langle \boldsymbol{p}_t, \boldsymbol{x} \rangle = h_{0:t-1}(\boldsymbol{x}) + \langle \boldsymbol{g}_t, \boldsymbol{x} \rangle + \langle \boldsymbol{g}_t^I, \boldsymbol{x} \rangle = \phi_1(\boldsymbol{x}) + \underbrace{\langle \boldsymbol{g}_t - \tilde{\boldsymbol{g}}_t, \boldsymbol{x} \rangle}_{\psi(\boldsymbol{x})}.$$

Under these definitions, we indeed have that

$$\begin{aligned} \boldsymbol{x}_t &= \arg\min_{\boldsymbol{x}} h_{0:t-1}(\boldsymbol{x}) + \tilde{f}_t(\boldsymbol{x}) \\ &\overset{\text{(Lem. 4A.1)}}{=} \arg\min_{\boldsymbol{x}} h_{0:t-1}(\boldsymbol{x}) + \langle \tilde{\boldsymbol{g}}_t, \boldsymbol{x} \rangle + \langle \boldsymbol{g}_t^I, \boldsymbol{x} \rangle = \arg\min_{\boldsymbol{x}} \phi_1(\boldsymbol{x}) = \boldsymbol{y}_1. \end{aligned}$$

and thus selecting $\boldsymbol{y}'$ of Lemma 4A.7 as $\boldsymbol{y}_t$, the result of the same (Lemma 4A.7) gives:

$$h_{0:t-1}(\boldsymbol{x}_t) + \langle \boldsymbol{p}_t, \boldsymbol{x}_t \rangle - h_{0:t-1}(\boldsymbol{y}_t) - \langle \boldsymbol{p}_t, \boldsymbol{y}_t \rangle \leq \frac{1}{2}\|\boldsymbol{g}_t - \tilde{\boldsymbol{g}}_t\|_{t-1,*}^2 \ , \tag{4A.9}$$

noticing that $\partial\psi(\boldsymbol{x}_t) = \boldsymbol{g}_t - \tilde{\boldsymbol{g}}_t$. □

From (4A.9) and (4A.8), we have that

$$\textbf{(I)} \leq \min\left(\frac{1}{2}\|\boldsymbol{g}_t - \tilde{\boldsymbol{g}}_t\|^2_{t-1,*}, 2R\epsilon_t\right) \tag{4A.10}$$

4A.2.3. Bounding the Second Part (II):

The second part does not have the structure exploited in the first one ($\boldsymbol{u}_t$ and $\boldsymbol{u}_{t+1}$ are arbitrary). Hence, we must resort to the strong convexity property to obtain

$$\begin{aligned} h_{0:t}(\boldsymbol{u}_{t+1}) - h_{0:t}(\boldsymbol{u}_t) &\leq \langle \boldsymbol{q}_t, \boldsymbol{u}_{t+1} - \boldsymbol{u}_t\rangle - \frac{\sigma_{1:t}}{2}\|\boldsymbol{u}_{t+1} - \boldsymbol{u}_t\|^2 \\ &\leq \|\boldsymbol{q}_t\|\|\boldsymbol{u}_{t+1} - \boldsymbol{u}_t\| - \frac{\sigma_{1:t}}{2}\|\boldsymbol{u}_{t+1} - \boldsymbol{u}_t\|^2, \end{aligned}$$

where

$$\boldsymbol{q}_t \in \partial h_{0:t}(\boldsymbol{u}_{t+1}).$$

Nonetheless, we will still exploit problem properties to bound $\|\boldsymbol{q}_t\|$:

$$\partial h_{0:t}(\boldsymbol{x}) = \boldsymbol{p}_{1:t} + \sum_{\tau=1}^{t} \sigma_\tau \boldsymbol{x} + \mathcal{N}_{\mathcal{X}}(\boldsymbol{x}),$$

which gives

$$\boldsymbol{q}_t = \boldsymbol{p}_{1:t} + \sum_{\tau=1}^{t} \sigma_\tau \boldsymbol{u}_{t+1},$$

where we chose the 0 vector from $\mathcal{N}_{\mathcal{X}}(\boldsymbol{u}_{t+1})$. Hence, the length of $\boldsymbol{q}_t$ satisfies:

$$\|\boldsymbol{q}_t\| \leq \|\boldsymbol{p}_{1:t}\| + \sum_{\tau=1}^{t} \sigma_\tau \|\boldsymbol{u}_{t+1}\| = \|\boldsymbol{p}_{1:t}\| + \sigma_{1:t}\|\boldsymbol{u}_{t+1}\| \leq \|\boldsymbol{p}_{1:t}\| + R\sigma_{1:t}$$

Overall

$$h_{0:t}(\boldsymbol{u}_{t+1}) - h_{0:t}(\boldsymbol{u}_t) \leq \left(\|\boldsymbol{p}_{1:t}\| + R\sigma_{1:t}\right)\|\boldsymbol{u}_{t+1} - \boldsymbol{u}_t\| - \frac{\sigma_{1:t}}{2}\|\boldsymbol{u}_{t+1} - \boldsymbol{u}_t\|^2 \tag{4A.11}$$

From (4A.11) it follows that

$$\textbf{(II)} \leq \left(\|\boldsymbol{p}_{1:t}\| + R\sigma_{1:t}\right)\|\boldsymbol{u}_{t+1} - \boldsymbol{u}_t\| - \frac{\sigma_{1:t}}{2}\|\boldsymbol{u}_{t+1} - \boldsymbol{u}_t\|^2 \tag{4A.12}$$

$$\leq \left(\|\boldsymbol{p}_{1:t}\| + R\sigma_{1:t}\right)\|\boldsymbol{u}_{t+1} - \boldsymbol{u}_t\| \tag{4A.13}$$

Note that the negative term in (4A.12) admits a tight lower bound of 0 and will therefore be omitted. Next, we derive a bound on the state vector's length $\|\boldsymbol{p}_{1:t}\|$.

Lemma 4A.3. *(Optimistically Bounded State) Let $\{\boldsymbol{p}_t\}_{t=1}^T$ be a sequence of vectors such that each $\boldsymbol{p}_t = \boldsymbol{g}_t + \boldsymbol{g}_t^I$, where $\boldsymbol{g}_t \in \partial f_t(\boldsymbol{x}_t)$, and $\boldsymbol{g}t^I \in \partial I_{\mathcal{X}}(\boldsymbol{x}_t)$ is chosen according to the construction in (4.5). That is, $\boldsymbol{p}_t$ corresponds to the linearization of the composite function $f_t(\cdot) + I_{\mathcal{X}}(\cdot)$ around $\boldsymbol{x}_t$. Then, for any t, the following holds:*

$$\|\boldsymbol{p}_{1:t}\| \leq R\sigma_{1:t-1} + \epsilon_t.$$

Proof. The proof of this lemma was stated in the paper. □

4A.3. Missing Proofs for Section 4.4

Recall the problem parameters described by the following settings:

Settings 1: Let $\mathcal{X} \subset \mathbb{R}^d$ be a compact, convex set such that $\|\boldsymbol{x}\| \leq R$ for all $\boldsymbol{x} \in \mathcal{X}$. Let $\{f_t(\cdot), \tilde{f}_t(\cdot)\}_{t=1}^T$ be any sequence of L-Lipschitz convex functions, and define the following quantities.

$$\text{The cumulative-squared prediction error: } E_T \doteq \sum_{t=1}^{T} \epsilon_t^2, \quad \epsilon_t \doteq \|\boldsymbol{g}_t - \tilde{\boldsymbol{g}}_t\|.$$

$$\text{The path length: } P_T \doteq \sum_{t=1}^{T-1} \|\boldsymbol{u}_{t+1} - \boldsymbol{u}_t\|$$

$$\text{The prediction-weighted path length: } H_T \doteq \sum_{t=1}^{T-1} \epsilon_t \|\boldsymbol{u}_{t+1} - \boldsymbol{u}_t\|$$

4A.3.1. Proof of Theorem 4.1

Consider the following regularization strategy

$$\sigma = \frac{1}{4R}, \text{ and } \sigma_1 = \sigma\epsilon_1, \ \sigma_t = \sigma\left(\sqrt{E_t} - \sqrt{E_{t-1}}\right), \forall t \geq 2. \tag{4A.14}$$

Theorem 4.1. *Under Settings 1, Alg. 10 run with the regularization strategy in* (4A.14) *produces points* $\{\boldsymbol{x}_t\}_{t=1}^T$ *such that, for any* T*, the dynamic regret* $\mathcal{R}_T$ *satisfies:*

$$\mathcal{R}_T \leq \left(5.8R + (1/2)P_T\right)\sqrt{E_T} + H_T = \mathcal{O}\left((1+P_T)\sqrt{E_T}\right).$$

Proof. We begin by substituting the bounds of **(I)** and **(II)**, from (4A.10) and (4A.13), respectively, back in the result of Lemma 4.5:

$$\begin{aligned}
&\mathcal{R}_T \\
&\leq \sum_{t=1}^{T}\left(\min\left(\frac{1}{2}\|\boldsymbol{g}_t - \tilde{\boldsymbol{g}}_t\|_{t-1,*}^2, 2R\epsilon_t\right) + r_t(\boldsymbol{u}_t)\right) \ + \ \sum_{t=1}^{T-1}\left(\left(R\sigma_{1:t} + \|\boldsymbol{p}_{1:t}\|\right)\|\boldsymbol{u}_{t+1} - \boldsymbol{u}_t\|\right) \\
&\overset{(a)}{\leq} \sum_{t=1}^{T}\left(\min\left(\frac{1}{2}\|\boldsymbol{g}_t - \tilde{\boldsymbol{g}}_t\|_{t-1,*}^2, 2R\epsilon_t\right) + r_t(\boldsymbol{u}_t)\right) \ + \ \sum_{t=1}^{T-1}\left(\left(2R\sigma_{1:t} + \epsilon_t\right)\|\boldsymbol{u}_{t+1} - \boldsymbol{u}_t\|\right) \\
&\overset{(b)}{\leq} \sum_{t=1}^{T}\min\left(\frac{1}{2}\|\boldsymbol{g}_t - \tilde{\boldsymbol{g}}_t\|_{t-1,*}^2, 2R\epsilon_t\right) + \frac{R^2}{2}\sigma_{1:T} \ + \ \sum_{t=1}^{T-1}\left(\left(2R\sigma_{1:t} + \epsilon_t\right)\|\boldsymbol{u}_{t+1} - \boldsymbol{u}_t\|\right) \\
&= \sum_{t=1}^{T}\min\left(\frac{\epsilon_t^2}{2\sigma_{0:t-1}}, 2R\epsilon_t\right) + \frac{R^2}{2}\sigma_{1:T} \ + \ \sum_{t=1}^{T-1}\left(\left(2R\sigma_{1:t} + \epsilon_t\right)\|\boldsymbol{u}_{t+1} - \boldsymbol{u}_t\|\right)
\end{aligned}$$

$$= \sum_{t=1}^{T} \min\left(\frac{\epsilon_t^2}{2\sigma\sqrt{E_{t-1}}}, 2R\epsilon_t\right) + \frac{R^2}{2}\sigma\sqrt{E_T} \; + \; \sum_{t=1}^{T-1}\left(\left(2R\sigma\sqrt{E_t} + \epsilon_t\right)\|\boldsymbol{u}_{t+1} - \boldsymbol{u}_t\|\right)$$

$$\overset{(c)}{\leq} 4\sqrt{2}R\sqrt{E_T} + \frac{R}{8}\sqrt{E_T} \; + \; \sum_{t=1}^{T-1}\frac{1}{2}\sqrt{E_t}\|\boldsymbol{u}_{t+1} - \boldsymbol{u}_t\| + H_T$$

$$\overset{(d)}{\leq} 5.8R\sqrt{E_T} \; + \; \frac{1}{2}\sqrt{E_{T-1}}\sum_{t=1}^{T-1}\|\boldsymbol{u}_{t+1} - \boldsymbol{u}_t\| + H_T$$

$$= 5.8R\sqrt{E_T} \; + \; \frac{1}{2}\sqrt{E_T}P_T + H_T$$

where (a) follows by Lemma 4.7 which bounds $\|\boldsymbol{p}_{1:t}\|$, and by the fact that $\sigma_{1:t-1} \leq \sigma_{1:t}$, (b) by bounding each $\|\boldsymbol{u}_t\|$ in $r_t(\boldsymbol{u}_t)$ by R, (c) by Lemma 4A.9, which is a common tool to bound the sum of a decreasing function (on E_{t-1}), with the choice of $\sigma = \frac{1}{4R}$, and (d) by the fact that $\sqrt{E_t}$ are non-decreasing. □

Remark 4A.3. On the growth rate of the hybrid term H_T.

Note that by Cauchy-Schwarz, we have that

$$H_T = \sum_{t=1}^{T-1}\epsilon_t\|\boldsymbol{u}_{t+1} - \boldsymbol{u}_t\| \leq \sqrt{\sum_{t=1}^{T-1}\epsilon_t^2}\sqrt{\sum_{t=1}^{T-1}\|\boldsymbol{u}_{t+1} - \boldsymbol{u}_t\|^2}$$
$$\leq \sqrt{2R}\sqrt{\sum_{t=1}^{T-1}\epsilon_t^2}\sqrt{\sum_{t=1}^{T-1}\|\boldsymbol{u}_{t+1} - \boldsymbol{u}_t\|} = \sqrt{2R}\sqrt{E_{T-1}}\sqrt{P_T}.$$

Hence, H_T is also $\mathcal{O}\left(\sqrt{P_T E_T}\right)$.

4A.3.2. Proof of Theorem 4.2

Consider the following regularization strategy:

$$\sigma = \frac{1}{2\sqrt{2RP'_T}}, \text{ where } P'_T \text{ is the augmented path length:} \quad P'_T \doteq 2R + P_T.$$
$$\sigma_1 = \sigma\epsilon_1, \; \sigma_t = \sigma\left(\sqrt{E_t} - \sqrt{E_{t-1}}\right), \forall t \geq 2. \tag{4A.15}$$

Theorem 4.2. *Under Settings 1, Alg. 10 run with the regularization strategy in* (4A.15) *produces points* $\{\boldsymbol{x}_t\}_{t=1}^T$ *such that, for any* T*, the dynamic regret* $\mathcal{R}_T$ *satisfies:*

$$\mathcal{R}_T \leq \left(4\sqrt{2R^2 + P_T} + \frac{R}{8} + \sqrt{\frac{RP_T}{2}}\right)\sqrt{E_T} + H_T \; = \; \mathcal{O}\left((1 + \sqrt{P_T})\sqrt{E_T}\right).$$

Proof. Similarly to Theorem 1, we begin by substituting the bounds of **(I)** and **(II)**, from (4A.10) and (4A.13), respectively, back in the result of Lemma 4.5:

$\mathcal{R}_T$

4A

$$\leq \sum_{t=1}^{T}\left(\min\left(\frac{1}{2}\|\boldsymbol{g}_t-\tilde{\boldsymbol{g}}_t\|^2_{t-1,*},2R\epsilon_t\right)+r_t(\boldsymbol{u}_t)\right)\ +\ \sum_{t=1}^{T-1}\left((R\sigma_{1:t}+\|\boldsymbol{p}_{1:t}\|)\,\|\boldsymbol{u}_{t+1}-\boldsymbol{u}_t\|\right)$$

$$\overset{(a)}{\leq} \sum_{t=1}^{T}\left(\min\left(\frac{1}{2}\|\boldsymbol{g}_t-\tilde{\boldsymbol{g}}_t\|^2_{t-1,*},2R\epsilon_t\right)+r_t(\boldsymbol{u}_t)\right)\ +\ \sum_{t=1}^{T-1}\left((2R\sigma_{1:t}+\epsilon_t)\,\|\boldsymbol{u}_{t+1}-\boldsymbol{u}_t\|\right)$$

$$\overset{(b)}{\leq} \sum_{t=1}^{T}\min\left(\frac{1}{2}\|\boldsymbol{g}_t-\tilde{\boldsymbol{g}}_t\|^2_{t-1,*},2R\epsilon_t\right)+\frac{R^2}{2}\sigma_{1:T}\ +\ \sum_{t=1}^{T-1}\left((2R\sigma_{1:t}+\epsilon_t)\,\|\boldsymbol{u}_{t+1}-\boldsymbol{u}_t\|\right)$$

$$= \sum_{t=1}^{T}\min\left(\frac{\epsilon_t^2}{2\sigma_{0:t-1}},2R\epsilon_t\right)+\frac{R^2}{2}\sigma_{1:T}\ +\ \sum_{t=1}^{T-1}\left((2R\sigma_{1:t}+\epsilon_t)\,\|\boldsymbol{u}_{t+1}-\boldsymbol{u}_t\|\right)$$

$$= \sum_{t=1}^{T}\min\left(\frac{\epsilon_t^2}{2\sigma\sqrt{E_{t-1}}},2R\epsilon_t\right)+\frac{R^2}{2}\sigma\sqrt{E_T}\ +\ \sum_{t=1}^{T-1}\left(\left(2R\sigma\sqrt{E_t}+\epsilon_t\right)\|\boldsymbol{u}_{t+1}-\boldsymbol{u}_t\|\right)$$

$$\overset{(c)}{\leq} 4\sqrt{R}\sqrt{P'_TE_T}+\frac{R^2}{4\sqrt{2RP'_T}}\sqrt{E_T}\ +\ \sum_{t=1}^{T-1}\left(\left(\frac{R}{\sqrt{2RP'_T}}\sqrt{E_t}+\epsilon_t\right)\|\boldsymbol{u}_{t+1}-\boldsymbol{u}_t\|\right)$$

$$\overset{(d)}{\leq} 4\sqrt{R}\sqrt{P'_TE_T}+\frac{R}{8}\sqrt{E_T}\ +\ \sqrt{\frac{R}{2}}\sum_{t=1}^{T-1}\sqrt{E_t}\frac{\|\boldsymbol{u}_{t+1}-\boldsymbol{u}_t\|}{\sqrt{P_T}}+H_T$$

$$\overset{(e)}{\leq} 4\sqrt{R}\sqrt{P'_TE_T}+\frac{R}{8}\sqrt{E_T}\ +\ \sqrt{\frac{R}{2}}\sqrt{E_{T-1}}\frac{P_T}{\sqrt{P_T}}+H_T$$

$$\leq 4\sqrt{R}\sqrt{P'_TE_T}+\frac{R}{8}\sqrt{E_T}\ +\ \sqrt{\frac{R}{2}}\sqrt{E_TP_T}+H_T$$

$$\leq \left(4\sqrt{RP'_T}+\frac{R}{8}+\sqrt{\frac{R}{2}}\sqrt{P_T}\right)\sqrt{E_T}+H_T$$

$$\leq \left(4\sqrt{R(2R+P_T)}+\frac{R}{8}+\sqrt{\frac{R}{2}}\sqrt{P_T}\right)\sqrt{E_T}+H_T$$

$$= \mathcal{O}\left((1+\sqrt{P_T})\sqrt{E_T}\right),$$

where (a) follows by Lemma 4.7 which bounds $\|\boldsymbol{p}_{1:t}\|$, and by $\sigma_{1:t-1}\leq\sigma_{1:t}$, (b) by bounding each $\|\boldsymbol{u}_t\|$ in $r_t(\boldsymbol{u}_t), t\leq T$ by R, (c) by lemma 4A.9, with the choice of $\sigma=\frac{1}{2\sqrt{2RP'_T}}$, (note that this choice still satisfies the Lemma's condition since $P'_T\geq 2R$),(d) also used that used $2R\leq P'_T$, (e) used that E_t is non-decreasing, and finally the $\mathcal{O}(\cdot)$ expression follows from Remark 4A.3. □

Remark 4A.4. On guaranteeing $\sqrt{P_TE_T},\forall u_t$.

To obtain a minimax bound that holds uniformly over all comparator sequences (i.e., without assuming prior knowledge of their path length, and without assuming that they are observable online), one can instantiate $\Theta(\log T)$ sub-learners, each with a halving σ starting from $1/\sqrt{T}$. Then, using the meta-learner of [168], the minimax bound can be recovered. The gist of this approach is that eventually $\exists$ an expert i such that $\forall P_T, \sigma^{(i)} \geq 1/\sqrt{P_T} \geq 1/2\sigma^{(i)}$.

4A.3.3. Proof of Theorem 4.3

Define the augmented seen path length at t as:

$$P'_t \doteq 2R + P_t = 2R + \sum_{\tau=1}^{t-1} \|\boldsymbol{u}_{\tau+1} - \boldsymbol{u}_\tau\|.$$

and consider the following regularization strategy

$$\sigma = \frac{1}{2\sqrt{2R}}, \quad \sigma_1 = \frac{\sigma\epsilon_1}{\sqrt{P'_1}}, \quad \sigma_t = \sigma \max\left(0, \sqrt{\frac{E_t}{P'_t}} - \sqrt{\frac{E_{t-1}}{P'_{t-1}}}\right), \forall t \geq 2. \tag{4A.16}$$

Theorem 4.3. *Under Settings 1, Alg. 10 run with the regularization strategy in* (4A.16) *produces points* $\{\boldsymbol{x}_t\}_{t=1}^T$ *such that, for any* T*, the dynamic regret* $\mathcal{R}_T$ *satisfies:*

$$\mathcal{R}_T \leq 5.5\sqrt{R}\sqrt{E_T P'_T} + H_T + \sqrt{R/2}\ A_T = \mathcal{O}\left((1+\sqrt{P_T})\sqrt{E_T} + A_T\right) \quad \textit{where}$$

$$A_T \doteq \sum_{t=1}^{T} \sum_{\tau \in [t]^+} \left(\sqrt{\frac{E_{\tau-1}}{P'_{t-1}}} - \sqrt{\frac{E_\tau}{P'_\tau}}\right) \|\boldsymbol{u}_{t+1} - \boldsymbol{u}_t\|,$$

$$\textit{with} \quad [t]^+ = \left\{2 \leq \tau \leq t \;\middle|\; \sqrt{\frac{E_{\tau-1}}{P'_{\tau-1}}} - \sqrt{\frac{E_\tau}{P'_\tau}} \geq 0\right\}.$$

Before proceeding to prove the theorem, we will make use of the following two lemmas, which we present independently to streamline the presentation.

Lemma 4A.5. *The squared norm* $\|\boldsymbol{g}_t - \tilde{\boldsymbol{g}}_t\|^2_{t-1} = \sigma_{0:t-1}\|\boldsymbol{g}_t - \tilde{\boldsymbol{g}}_t\|^2 = \sigma_{0:t-1}\epsilon_t^2$ *can be written as:*

$$\|\boldsymbol{g}_t - \tilde{\boldsymbol{g}}_t\|^2_{0,*} = 0,$$

$$\|\boldsymbol{g}_t - \tilde{\boldsymbol{g}}_t\|^2_{t-1,*} \leq \frac{1}{\sigma}\sqrt{\frac{P'_{t-1}}{E_{t-1}}}\ \epsilon_t^2, \qquad \forall t \geq 1$$

Proof. Recall first that the dual norm of $\|\cdot\|_{t-1,*}$ is $\frac{1}{\sqrt{\sigma_{0:t-1}}}\|\cdot\|$. The first part is immediate from $\|\boldsymbol{x}\|_{0,*} = \sqrt{\frac{E_0}{P'_0}}\|\boldsymbol{x}\| = 0$, since $E_0 = 0$ and $P'_0 = 2R$. For the second part:

$$\|\boldsymbol{g}_t - \tilde{\boldsymbol{g}}_t\|^2_{t-1,*} = \frac{\|\boldsymbol{g}_t - \tilde{\boldsymbol{g}}_t\|^2}{\sigma_{1:t-1}} = \frac{\epsilon_t^2}{\frac{\sigma\epsilon_1}{\sqrt{P'_1}} + \sum_{\tau=2}^{t-1} \sigma \max\left(0,\ \sqrt{\frac{E_\tau}{P'_\tau}} - \sqrt{\frac{E_{\tau-1}}{P'_{\tau-1}}}\right)}$$

$$\leq \frac{\epsilon_t^2}{\frac{\sigma\epsilon_1}{\sqrt{P_1'}} + \sum_{\tau=2}^{t-1} \sigma\left(\sqrt{\frac{E_\tau}{P_\tau'}} - \sqrt{\frac{E_{\tau-1}}{P_{\tau-1}'}}\right)} = \frac{\epsilon_t^2}{\sigma\sqrt{\frac{E_{t-1}}{P_{t-1}'}}},$$

where the inequality follows by dropping the max in the denominator. □

Lemma 4A.6. *The strong convexity parameter $\sigma_{1:t}$ in* (4A.16) *can be written as the cumulative term $\sqrt{\frac{E_t}{P_t'}}$ plus a corrective term:*

$$\sigma_{1:t} \;=\; \sigma\left(\sqrt{\frac{E_t}{P_t'}} + \sum_{\tau\in[t]^+} \sqrt{\frac{E_{\tau-1}}{P_{\tau-1}'}} - \sqrt{\frac{E_\tau}{P_\tau'}}\right).$$

Proof.

$$\begin{aligned}\sigma_{1:t} &= \frac{\sigma\epsilon_1}{\sqrt{P_1'}} + \sigma\sum_{\tau=2}^{t} \max\left(0,\; \sqrt{\frac{E_\tau}{P_\tau'}} - \sqrt{\frac{E_{\tau-1}}{P_{\tau-1}'}}\right) \\ &= \sigma\left(\frac{\epsilon_1}{\sqrt{P_1'}} + \sum_{\tau=2}^{t} \sqrt{\frac{E_\tau}{P_\tau'}} - \sqrt{\frac{E_{\tau-1}}{P_{\tau-1}'}}\right) + \sigma\sum_{\tau\in[t]^+} \sqrt{\frac{E_{\tau-1}}{P_{\tau-1}'}} - \sqrt{\frac{E_\tau}{P_\tau'}},\end{aligned}$$

where the last equality holds from the definition of $[t]^+$. That is, for the slot τ when the max evaluates to 0, we write it as $\left(\sqrt{\frac{E_\tau}{P_\tau'}} - \sqrt{\frac{E_{\tau-1}}{P_{\tau-1}'}}\right) + \left(\sqrt{\frac{E_{\tau-1}}{P_{\tau-1}'}} - \sqrt{\frac{E_\tau}{P_\tau'}}\right)$. Now, by telescoping

$$\begin{aligned}\sigma_{1:t} &= \sigma\left(\frac{\epsilon_1}{\sqrt{P_1'}} + \sqrt{\frac{E_t}{P_t'}} - \sqrt{\frac{E_1}{P_1'}}\right) + \sigma\sum_{\tau\in[t]^+} \sqrt{\frac{E_{\tau-1}}{P_{\tau-1}'}} - \sqrt{\frac{E_\tau}{P_\tau'}} \\ &= \sigma\sqrt{\frac{E_t}{P_t'}} + \sigma\sum_{\tau\in[t]^+} \sqrt{\frac{E_{\tau-1}}{P_{\tau-1}'}} - \sqrt{\frac{E_\tau}{P_\tau'}}.\end{aligned}$$

□

Proof of Theorem 4.3. We begin by substituting the bounds of **(I)** and **(II)**, in (4A.10) and (4A.13), respectively, back in the result of Lemma 4.5:

$$\begin{aligned}&\mathcal{R}_T \\ &\leq \sum_{t=1}^{T}\left(\min\left(\frac{1}{2}\|\boldsymbol{g}_t - \tilde{\boldsymbol{g}}_t\|_{t-1,*}^2, 2R\epsilon_t\right) + r_t(\boldsymbol{u}_t)\right) + \sum_{t=1}^{T-1}\left(\left(R\sigma_{1:t} + \|\boldsymbol{p}_{1:t}\|\right)\|\boldsymbol{u}_{t+1} - \boldsymbol{u}_t\|\right) \\ &\overset{(a)}{\leq} \sum_{t=1}^{T}\left(\min\left(\frac{1}{2}\|\boldsymbol{g}_t - \tilde{\boldsymbol{g}}_t\|_{t-1,*}^2, 2R\epsilon_t\right) + r_t(\boldsymbol{u}_t)\right) + \sum_{t=1}^{T-1}\left(\left(2R\sigma_{1:t} + \epsilon_t\right)\|\boldsymbol{u}_{t+1} - \boldsymbol{u}_t\|\right)\end{aligned}$$

$$
\begin{aligned}
&\overset{(b)}{\leq} \sum_{t=1}^{T} \min\left(\frac{1}{2}\|\boldsymbol{g}_t - \tilde{\boldsymbol{g}}_t\|^2_{t-1,*}, 2R\epsilon_t\right) + \frac{R^2}{2}\sigma_{1:T} + \sum_{t=1}^{T-1} 2R\sigma_{1:t}\|\boldsymbol{u}_{t+1} - \boldsymbol{u}_t\| + H_T \\
&\overset{(c)}{\leq} \sum_{t=1}^{T} \min\left(\frac{\sqrt{P'_{t-1}}\epsilon_t^2}{2\sigma\sqrt{E_{t-1}}}, 2R\epsilon_t\right) + \sum_{t=1}^{T} 2R\sigma_{1:t}\|\boldsymbol{u}_{t+1} - \boldsymbol{u}_t\| + H_T \\
&\overset{(d)}{\leq} \sum_{t=1}^{T} \min\left(\frac{\sqrt{P'_{t-1}}\epsilon_t^2}{2\sigma\sqrt{E_{t-1}}}, 2R\epsilon_t\right) + 2R\sigma\sum_{t=1}^{T}\sqrt{\frac{E_t}{P'_t}}\|\boldsymbol{u}_{t+1} - \boldsymbol{u}_t\| + H_T \\
&\quad + 2R\sigma \underbrace{\sum_{t=1}^{T}\sum_{\tau\in[t]^+}\left(\sqrt{\frac{E_{\tau-1}}{P'_{t-1}}} - \sqrt{\frac{E_\tau}{P'_\tau}}\right)\|\boldsymbol{u}_{t+1} - \boldsymbol{u}_t\|}_{\doteq A_T} \\
&\overset{(e)}{\leq} \sum_{t=1}^{T}\sqrt{P'_T}\min\left(\frac{\epsilon_t^2}{2\sigma\sqrt{E_{t-1}}}, \frac{2R}{\sqrt{P'_T}}\epsilon_t\right) + 2R\sigma\sum_{t=1}^{T}\sqrt{\frac{E_t}{P'_t}}\|\boldsymbol{u}_{t+1} - \boldsymbol{u}_t\| + H_T + 2R\sigma A_T \\
&\overset{(f)}{\leq} \sum_{t=1}^{T}\sqrt{P'_T}\min\left(\frac{\epsilon_t^2}{2\sigma\sqrt{E_{t-1}}}, \sqrt{2R}\epsilon_t\right) + 2R\sigma\sqrt{E_T}\sum_{t=1}^{T}\frac{\|\boldsymbol{u}_{t+1} - \boldsymbol{u}_t\|}{\sqrt{P'_t}} + H_T + 2R\sigma A_T \\
&\overset{(g)}{\leq} 4\sqrt{R}\sqrt{P'_T E_T} \;+\; \sqrt{\frac{R}{2}}\sqrt{E_T}\sum_{t=1}^{T}\frac{\|\boldsymbol{u}_{t+1} - \boldsymbol{u}_t\|}{\sqrt{2R + \sum_{\tau=1}^{t-1}\|\boldsymbol{u}_{\tau+1} - \boldsymbol{u}_\tau\|}} + H_T + \sqrt{\frac{R}{2}}A_T \\
&\overset{(h)}{\leq} 4\sqrt{R}\sqrt{P'_T E_T} \;+\; \sqrt{\frac{R}{2}}\sqrt{E_T}\sum_{t=1}^{T}\frac{\|\boldsymbol{u}_{t+1} - \boldsymbol{u}_t\|}{\sqrt{\sum_{\tau=1}^{t}\|\boldsymbol{u}_{\tau+1} - \boldsymbol{u}_\tau\|}} + H_T + \sqrt{\frac{R}{2}}A_T \\
&\overset{(i)}{\leq} 4\sqrt{R}\sqrt{P'_T E_T} \;+\; \sqrt{2R}\sqrt{E_T P'_T} + H_T + \sqrt{\frac{R}{2}}A_T \\
&= (4+\sqrt{2})\sqrt{R}\sqrt{P'_T E_T} \;+ H_T + \sqrt{\frac{R}{2}}A_T
\end{aligned}
$$

where (a) follows by Lemma 4.7 which bounds $\|\boldsymbol{p}_{1:t}\|$, and the fact that $\sigma_{1:t-1} \leq \sigma_{1:t}$, (b) by bounding each $\|\boldsymbol{u}_t\|$ in $r_t(\boldsymbol{u}_t), t \leq T$ by R, (c) by using Lemma 4A.5 for the first sum, and by selecting[2] $\boldsymbol{u}_{T+1}$ such that $\|\boldsymbol{u}_{T+1} - \boldsymbol{u}_T\| \geq \frac{R}{4}$, which allows us to append the $\frac{R^2}{2}\sigma_{1:T}$ term to the second sum as the summand with index T, (d) by using Lemma 4A.6 to re-write $\sigma_{1:t}$, (e) since P'_t is non-decreasing, (f) from $P'_t \geq 2R$, (g) by $\sigma = 1/2\sqrt{2R}$ and Lemma 4A.9, (h) by the fact that $2R \geq \|\boldsymbol{u}_{t+1} - \boldsymbol{u}_t\|, \forall t$, and finally (i) by Lemma 4A.8 with $a_t = \|\boldsymbol{u}_{t+1} - \boldsymbol{u}_t\|$. □

[2]Note that $\boldsymbol{u}_{T+1}$ does not affect the algorithm and hence we can set it without loss of generality

Bounding the A_T term

In this subsection, we show that the term A_T cannot be worse than the result of Theorem 1 in all cases.

$$
\begin{aligned}
A_T &\doteq \sum_{t=1}^{T} \sum_{\tau \in [t]^+} \left(\sqrt{\frac{E_{\tau-1}}{P'_{\tau-1}}} - \sqrt{\frac{E_\tau}{P'_\tau}} \right) \|\boldsymbol{u}_{t+1} - \boldsymbol{u}_t\| \\
&= \sum_{t=1}^{T} \left[\|\boldsymbol{u}_{t+1} - \boldsymbol{u}_t\| \sum_{\tau \in [t]^+} \left(\sqrt{\frac{E_{\tau-1}}{P'_{\tau-1}}} - \sqrt{\frac{E_\tau}{P'_\tau}} \right) \right] \\
&\overset{(a)}{\leq} \sum_{t=1}^{T} \left[\|\boldsymbol{u}_{t+1} - \boldsymbol{u}_t\| \sum_{\tau \in [t]^+} \left(\sqrt{\frac{E_\tau}{P'_{\tau-1}}} - \sqrt{\frac{E_\tau}{P'_\tau}} \right) \right] \\
&= \sum_{t=1}^{T} \left[\|\boldsymbol{u}_{t+1} - \boldsymbol{u}_t\| \sum_{\tau \in [t]^+} \sqrt{E_\tau} \left(\sqrt{\frac{1}{P'_{\tau-1}}} - \sqrt{\frac{1}{P'_\tau}} \right) \right] \\
&\overset{(b)}{\leq} \sum_{t=1}^{T} \left[\|\boldsymbol{u}_{t+1} - \boldsymbol{u}_t\| \sqrt{E_t} \sum_{\tau \in [t]^+} \left(\sqrt{\frac{1}{P'_{\tau-1}}} - \sqrt{\frac{1}{P'_\tau}} \right) \right] \\
&\overset{(c)}{\leq} \sum_{t=1}^{T} \left[\|\boldsymbol{u}_{t+1} - \boldsymbol{u}_t\| \sqrt{E_t} \sum_{\tau=2}^{t} \left(\sqrt{\frac{1}{P'_{\tau-1}}} - \sqrt{\frac{1}{P'_\tau}} \right) \right] \\
&\overset{(d)}{=} \sum_{t=1}^{T} \left[\|\boldsymbol{u}_{t+1} - \boldsymbol{u}_t\| \sqrt{E_t} \left(\sqrt{\frac{1}{P'_1}} - \sqrt{\frac{1}{P'_t}} \right) \right] \\
&\overset{(e)}{=} \sum_{t=1}^{T} \left[\|\boldsymbol{u}_{t+1} - \boldsymbol{u}_t\| \sqrt{E_t} \sqrt{\frac{1}{P'_1}} \right] \\
&\overset{(f)}{\leq} \sqrt{\frac{E_T}{P'_1}} \sum_{t=1}^{T} \|\boldsymbol{u}_{t+1} - \boldsymbol{u}_t\| \\
&= \frac{1}{\sqrt{P'_1}} \sqrt{E_T} P'_T \;=\; \mathcal{O}(\sqrt{E_T}(P_T + 1)),
\end{aligned}
$$

where (a) is from $E_\tau \geq E_{\tau-1}$ for all τ by definition; (b) holds similarly because E_τ is non-decreasing on τ (or t); (c) holds because $P'_{\tau-1} \leq P'_\tau$ for all τ (for any slot t) and hence we can add additional positive terms and create the entire sum from $\tau = 2$ to $\tau = t$, instead of only the partial sum of terms in $[t]^+$; (d) holds by writing the telescoping sum; (e) holds by dropping the last term which is negative; and finally in (f) we used the fact that $E_T \geq E_t, \forall t$.

4A.3.4. Proof of Theorem 4.4

Consider the following regularization strategy

$$\sigma = \frac{1}{8R^2}; \quad \sigma_t = \sigma\delta_t$$
$$\delta_1 = \langle \boldsymbol{g}_1, \boldsymbol{x}_1\rangle - \min_{\boldsymbol{x}\in\mathcal{X}}\langle \boldsymbol{g}_1, \boldsymbol{x}\rangle;\ \delta_t = h_{0:t-1}(\boldsymbol{x}_t) + \langle \boldsymbol{p}_t, \boldsymbol{x}_t\rangle - \min_{\boldsymbol{x}\in\mathcal{X}}\left(h_{0:t-1}(\boldsymbol{x}) + \langle \boldsymbol{p}_t, x\rangle\right), \forall t \geq 2; \tag{4A.17}$$

Clarification on the term "recursive": In previous regularization strategies, the strong convexity at time t, $\sigma_{1:t}$, can be expressed in closed form as $\sigma_{1:t} = \sigma\sqrt{E_t}$. This follows from defining each σ_t to be exactly $\sigma(\sqrt{E_t} - \sqrt{E_{t-1}}) \leq \frac{\sigma\epsilon_t^2}{2\sqrt{E_{t-1}}}$. However, when σ_t is defined more generally as a scalar $\sigma\delta_t$, where δ_t can take any value in $[0, \sigma(\frac{\epsilon_t^2}{2\sqrt{E_{t-1}}})]$, we lose this compact form. Instead, $\sigma_{1:t}$ is now recursively defined as $\sigma_{1:t-1} + \sigma\delta_t$. As discussed, this ensures minimal regularization, impacting both the algorithm's behavior and the analysis.

Theorem 4.4. *Under Settings 1, Alg. 10 run with the regularization strategy in* (4A.17) *produces points* $\{\boldsymbol{x}_t\}_{t=1}^T$ *such that, for any* T*, the dynamic regret* $\mathcal{R}_T$ *satisfies:*

$$\begin{aligned}\mathcal{R}_T &\leq 1.1\ \delta_{1:T} + \sum_{t=1}^{T-1}\frac{1}{4R}\delta_{1:t}\|\boldsymbol{u}_{t+1} - \boldsymbol{u}_t\| + H_T \\ &\leq (3.7R + P_T)\sqrt{E_T} + H_T = \mathcal{O}\left((1+P_T)\sqrt{E_T}\right).\end{aligned}$$

Proof. We begin from the result of Lemma 4.5 but without substituting the upper bound on part **(I)**, in order to characterize it more tightly via the δ_t terms. For the term **(II)**, we use the upper bound from (4A.13):

$$\mathcal{R}_T \leq \sum_{t=1}^{T}\left(h_{0:t}(\boldsymbol{x}_t) - h_{0:t}(\boldsymbol{x}_{t+1}) + r_t(\boldsymbol{u}_t) - r_t(\boldsymbol{x}_t)\right) + \sum_{t=1}^{T-1}\left(\left(R\sigma_{1:t} + \|\boldsymbol{p}_{1:t}\|\right)\|\boldsymbol{u}_{t+1} - \boldsymbol{u}_t\|\right)$$

$$\begin{aligned}&\overset{(a)}{\leq} \sum_{t=1}^{T}\left(h_{0:t}(\boldsymbol{x}_t) - r_t(\boldsymbol{x}_t) - h_{0:t-1}(\boldsymbol{x}_{t+1}) - r_t(\boldsymbol{x}_{t+1}) - \langle \boldsymbol{p}_t, \boldsymbol{x}_{t+1}\rangle + r_t(\boldsymbol{u}_t)\right) \\ &\quad + \sum_{t=1}^{T-1}\left(\left(2R\sigma_{1:t} + \epsilon_t\right)\|\boldsymbol{u}_{t+1} - \boldsymbol{u}_t\|\right) \\ &\overset{(b)}{\leq} \sum_{t=1}^{T}\left(h_{0:t-1}(\boldsymbol{x}_t) + \langle \boldsymbol{p}_t, \boldsymbol{x}_t\rangle - h_{0:t-1}(\boldsymbol{x}_{t+1}) - \langle \boldsymbol{p}_t, \boldsymbol{x}_{t+1}\rangle + r_t(\boldsymbol{u}_t)\right) \\ &\quad + \sum_{t=1}^{T-1}\left(\left(2R\sigma_{1:t} + \epsilon_t\right)\|\boldsymbol{u}_{t+1} - \boldsymbol{u}_t\|\right) \\ &\overset{(c)}{\leq} \sum_{t=1}^{T}\left(\delta_t + r_t(\boldsymbol{u}_t)\right) + \sum_{t=1}^{T-1}\left(\left(2R\sigma_{1:t} + \epsilon_t\right)\|\boldsymbol{u}_{t+1} - \boldsymbol{u}_t\|\right)\end{aligned}$$

$$\overset{(d)}{\leq} \delta_{1:T} + \frac{R^2\sigma}{2}\delta_{1:T} + \sum_{t=1}^{T-1} 2R\sigma_{1:t}\|\boldsymbol{u}_{t+1} - \boldsymbol{u}_t\| + H_T$$
$$\overset{(e)}{\leq} 1.1\ \delta_{1:T} + \sum_{t=1}^{T-1} \frac{1}{4R}\delta_{1:t}\|\boldsymbol{u}_{t+1} - \boldsymbol{u}_t\| + H_T \tag{4A.18}$$

where (a) follows by Lemma 4.7 which bounds $\|\boldsymbol{p}_{1:t}\|$, and by the fact that $\sigma_{1:t-1} \leq \sigma_{1:t}$, (b) by dropping the non-negative $-r_t(\boldsymbol{x}_{t+1})$, (c) by the definition of δ_t, (d) by bounding each $\|\boldsymbol{u}_t\|$ in $r_t(\boldsymbol{u}_t), t \leq T$ by R, (e) since $\sigma = 1/8R^2$.

Bounding the Recursion

Note for each δ_t we have that

$$\delta_t \leq 2R\epsilon_t.$$

The above follows from Lemma 4A.2. Specifically, note that δ_t is equal to the expression in (4A.7). In addition, we know from Lemma 4.6 that

$$\delta_t \leq \frac{\epsilon_t^2}{2\sigma_{1:t-1}},$$

by noticing that δ_t is the LHS in the inequality (4A.9). Substituting the strong convexity term $\sigma_{1:t}$ for the choices made in this section in (4A.17), we get

$$\delta_t \leq \frac{4R^2\epsilon_t^2}{\delta_{1:t-1}}.$$

Overall,

$$\delta_t \leq \min\left(2R\epsilon_t, \frac{4R^2\epsilon_t^2}{\delta_{1:t-1}}\right).$$

We can now invoke the auxiliary Lemma 4A.10 on the last term with $a_t \doteq 2R\epsilon_t$, $\Delta_t \doteq \delta_{1:t}$ to get that for any t,

$$\delta_{1:t} \leq 2\sqrt{3}R\sqrt{E_t}.$$

Going back to (4A.18), we get

$$\mathcal{R}_T \leq \frac{17\sqrt{3}}{8}R\sqrt{E_T} + \sum_{t=1}^{T-1}\frac{\sqrt{3}}{2}\sqrt{E_t}\|\boldsymbol{u}_{t+1} - \boldsymbol{u}_t\| + H_T$$
$$\leq \frac{17\sqrt{3}}{8}R\sqrt{E_T} + \frac{\sqrt{3}}{2}\sqrt{E_T}P_T + H_T = \mathcal{O}\left((1+P_T)\sqrt{E_T}\right).$$

□

4A.4. Auxiliary Lemmas

Lemmas/theorems here are (specialized) results from the literature with potentially modified notation.

Lemma 4A.7. *[112, Lemma 7] Given a convex function $\phi_1(\cdot)$ with a minimizer $\boldsymbol{y}_1 \doteq \arg\min_{\boldsymbol{y}} \phi_1(\boldsymbol{y})$ and a function $\phi_2(\cdot) = \phi_1(\cdot) + \psi(\cdot)$ that is 1-strongly convex w.r.t some norm $\|\cdot\|$. Then,*

$$\phi_2(\boldsymbol{y}_1) - \phi_2(\boldsymbol{y}') \leq \frac{1}{2}\|\boldsymbol{b}\|_*^2,$$

for any $\boldsymbol{y}'$, and any (sub)gradient $b \in \partial\psi(\boldsymbol{y}_1)$.

Lemma 4A.8. *[114, Lemma 3.5] For any $a_t \geq 0, \forall t \in [T]$*

$$\sum_{t=1}^{T} \frac{a_t}{\sqrt{a_{1:t}}} \leq 2\sqrt{\sum_{t=1}^{T} a_t},$$

with the convention that $\frac{0}{\sqrt{0}} \doteq 0$.

Lemma 4A.9. *[28, Section 7.6] Let σ be a positive real number that satisfy $\sigma \leq \frac{1}{2b}$, $b > 0$. Then, for any $a_t \geq 0, \forall t \in [T]$:*

$$\sum_{t=1}^{T} \min\left(\frac{a_t^2}{2\sigma\sqrt{\sum_{\tau=1}^{t-1} a_\tau}}, ba_t\right) \leq \frac{\sqrt{2}}{\sigma}\sqrt{\sum_{t=1}^{T} a_t^2}.$$

Proof. The proof of this lemma closely follows the argument presented in the cited section. However, due to the inclusion of the term b in the second part of the min, we adapt the proof accordingly to account for this difference.

$$\begin{aligned}
&\sum_{t=1}^{T} \min\left(\frac{a_t^2}{2\sigma\sqrt{\sum_{\tau=1}^{t-1} a_\tau^2}}, ba_t\right) = \sum_{t=1}^{T}\sqrt{\min\left(\frac{a_t^4}{4\sigma^2\sum_{\tau=1}^{t-1} a_\tau^2}, b^2a_t^2\right)} \\
&= \frac{1}{2}\sum_{t=1}^{T}\sqrt{\min\left(\frac{a_t^4}{\sigma^2\sum_{\tau=1}^{t-1} a_\tau^2}, 4b^2a_t^2\right)} \overset{(a)}{\leq} \frac{1}{2}\sum_{t=1}^{T}\sqrt{\frac{2}{\frac{\sigma^2\sum_{\tau=1}^{t-1} a_\tau^2}{a_t^4} + \frac{1}{4b^2a_t^2}}} \\
&= \frac{1}{2}\sum_{t=1}^{T}\sqrt{\frac{2}{\frac{4b^2\sigma^2\sum_{\tau=1}^{t-1} a_\tau^2 + a_t^2}{4b^2a_t^4}}} \overset{(b)}{\leq} \frac{1}{2}\sum_{t=1}^{T}\frac{2\sqrt{2}\, b\, a_t^2}{\sqrt{4b^2\sigma^2\sum_{\tau=1}^{t} a_\tau^2}} = \frac{\sqrt{2}}{2\sigma}\sum_{t=1}^{T}\frac{a_t^2}{\sqrt{\sum_{\tau=1}^{t} a_\tau^2}},
\end{aligned}$$

where (a) follows by $\min(a,b) \leq \frac{2}{1/a+1/b}$, and (b) used the assumption on σ to append a_t^2 to the sum. The result then follows by lemma 4A.8. □

Lemma 4A.10. *[164, Lemma 7] Let $a_t \geq 0$, $\forall t \in [T]$, and Δ_t be a sequence of positive numbers satisfying the recurrence*

$$\Delta_t = \Delta_{t-1} + \min\left(a_t, \frac{a_t^2}{\Delta_{t-1}}\right).$$

with $\Delta_0 \doteq 0$. Then, for any T, we have that

$$\Delta_T = \sqrt{3\sum_{t=1}^{T} a_t^2}.$$

4A.5. Comparison with the Literature

We summarize the key differences between our work and the relevant existing results on optimistic dynamic regret. The selected references represent the best known regret bounds (there are many other works that focus on different aspects, such as efficiency, unbounded domain, etc, but have the same structure as these bounds). It is important to note that some of these works derive bounds in terms of quantities other than the gradient prediction error, such as the extended "temporal variation" in [161] or the "comparator loss" in [169]. For the sake of consistency, we restrict the comparison to bounds involving E_T, choosing E_T in cases where a $\min(E_T, \cdot)$ term is present in the literature. Extending FTRL-based algorithms to handle comparator loss and temporal variations, dynamic regret bounds remain open. Lastly, in the comparison below, "best-case" refers to the scenario where $E_T = 0$, corresponding to perfect predictions.

Bounds without tuning for P_T:

- [160] Obtains a bound of $\mathcal{O}((P_T + 1)\sqrt{E_T + 1})$, which is $\mathcal{O}(P_T)$ in the best case.
- [161] Obtains a bound of $\mathcal{O}((P_T+1)\sqrt{D_T + 1})$, where $D_T \doteq \|\sum_{t=1}^{T} \nabla f_t(\boldsymbol{y}_{t-1}) - \nabla \tilde{f}_t(\boldsymbol{y}_{t-1})\|$. which is $\mathcal{O}(P_T)$ in the best case.
- [169] Obtains a bound $\mathcal{O}(\sqrt{(V_T + P_T + 1)(1 + P_T)})$, which is $\mathcal{O}(P_T)$ in the best case. Note that all P_T appear under the root. Hence, this bound still matches the optimal min-max bound of $\sqrt{T(1 + P_T)}$ in the worst case.
- This work achieves a dynamic regret bound of $\mathcal{R}_T = \mathcal{O}((1 + P_T)\sqrt{E_T}$, which is 0 in the best case.

Bounds with online P_T estimation:

- [160] Obtains a bound of $\mathcal{O}(\log(T)\sqrt{(P_T + 1)(E_T + 1)})$, which is $\mathcal{O}(\log(T)\sqrt{P_T})$ in the best case.
- [161] Obtains a bound of $\mathcal{O}(\sqrt{(P_T + 1)(D_T + \theta_T)})$, where θ_T is the sum of corrective terms θ_t, which are added to the learning rate to ensure monotonicity, and it holds that in the perfect *gradient* prediction case $\theta_T = \mathcal{O}(1 + P_T)$ and the regret bound is also $\mathcal{O}(P_T)$.

- [169] Maintains the same bounds since, as detailed in the main paper, this meta learning framework provides bounds that hold simultaneously for all P_T. The bound is $\mathcal{O}(\sqrt{(V_T + P_T + 1)(1 + P_T)})$, which is $\mathcal{O}(P_T)$ in the best case.
- This work achieves a dynamic regret bound of $\mathcal{O}((1+\sqrt{P_T})\sqrt{E_T}+A_T)$, which is 0 in the best case.

4A.6. Numerical Examples

In this section, we present numerical examples comparing the performance of three standard implementations of OCO algorithms across multiple non-stationary environments (sequences of cost functions). The algorithms considered are:

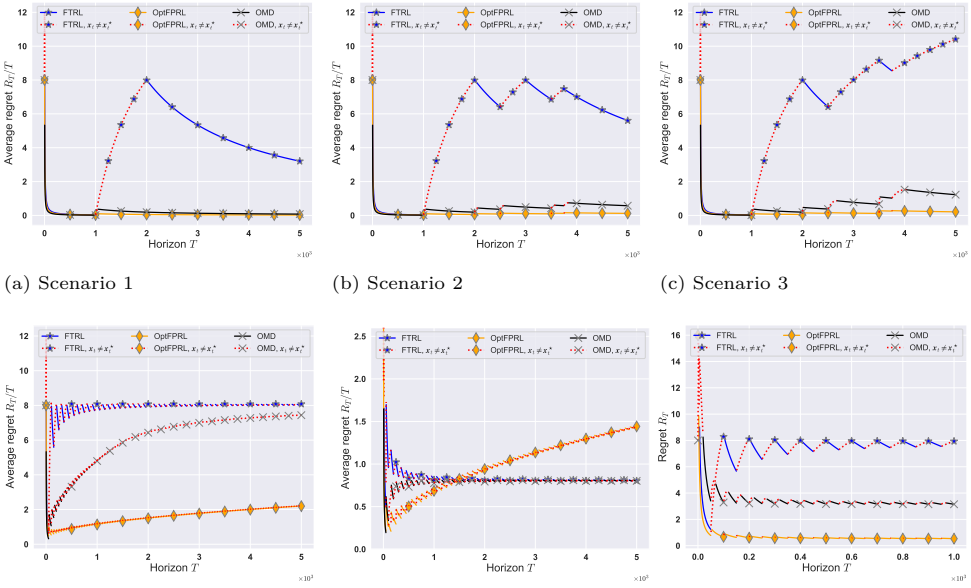
- FTRL with adaptive Euclidean regularization, which corresponds to a lazy projected Online Gradient Descent (OGD) but with Adagrad style tuning [112, Sec. 3.5]
- OMD with data-adaptive learning rates, which corresponds to a greedy projection OGD [28, Sec. 4.2].
- Our proposed algorithm, `OptFPRL`, with the vanilla tuning strategy (i.e., in Sec. 3.1).

The implementation code for the algorithms, along with the code to reproduce all experiments, is available at the following repository: [228]. Since the FTRL and OMD variants used in our experiments neither assume prior knowledge of P_T nor attempt to estimate it online, we compare them to `OptFPRL` using the tuning strategy described in Sec. 3.1 to ensure a fair evaluation. In scenarios where no predictions are used, the predicted functions fed to the algorithms are set to zero. This allows us to understand their performance initially, independent of prediction quality. Even without predictions, the proposed algorithm outperforms the two benchmarks in many scenarios, demonstrating its performance in dynamic environments.

Numerical setup. $\mathcal{X} \doteq \{\boldsymbol{x} \in \mathbb{R}^{16} | \|\boldsymbol{x}\| \leq 2\}$ $f_t(\boldsymbol{x}) = \langle \boldsymbol{c}_t, \boldsymbol{x} \rangle$, $T = 5000$, with

- Scenario 1: $\boldsymbol{c}_1, \ldots, \boldsymbol{c}_{1000} = -\mathbf{1}, \quad \boldsymbol{c}_{1000}, \ldots, \quad \boldsymbol{c}_{5000} = \mathbf{1}$.
- Scenario 2: $\boldsymbol{c}_1, \ldots, \boldsymbol{c}_{1000} = -\mathbf{1}, \quad \boldsymbol{c}_{2000}, \ldots, \boldsymbol{c}_{2500} = -\mathbf{1}, \quad \boldsymbol{c}_{3500}, \ldots, \quad \boldsymbol{c}_{3750} = -\mathbf{1}$, $\boldsymbol{c}_t = \mathbf{1}$ otherwise.
- Scenario 3: $\boldsymbol{c}_1, \ldots, \boldsymbol{c}_{1000} = -\mathbf{1}, \quad \boldsymbol{c}_{2000}, \ldots, \boldsymbol{c}_{2500} = -\mathbf{5}, \quad \boldsymbol{c}_{3500}, \ldots, \quad \boldsymbol{c}_{3750} = -\mathbf{10}$, $c_t = \mathbf{1}$ otherwise.
- Scenario 4: $\boldsymbol{c}_t$ alternates between $\mathbf{1}$ and -$\mathbf{1}$ every 50 steps.
- Scenario 5: $\boldsymbol{c}_t$ alternates between $\mathbf{1}$ and -$\mathbf{0.1}$ every 50 steps.
- Scenario 6: $\boldsymbol{c}_t$ alternates between $\mathbf{1}$ and -$\mathbf{1}$ every 50 steps; Predictions $\tilde{\boldsymbol{c}}_t = \boldsymbol{c}_t - \frac{\boldsymbol{c}_t}{0.1t}, \forall t \in [T]$

The observed behavior across different scenarios in the above figure aligns with theoretical expectations. In Scenario 1, where the cost function shifts at $t = 1000$, standard FTRL struggles to adapt due to its reliance on accumulating all past costs. This inertia makes it slow to respond and leads to continued suboptimal actions until $t = 2000$, resulting in high regret. In contrast, OMD reacts immediately to the



(a) Scenario 1 (b) Scenario 2 (c) Scenario 3

(d) Scenario 4 (e) Scenario 5 (f) Scenario 6

Figure 4A.1: Average dynamic regret over time across various non-stationary scenarios. Dashed lines indicate time slots where the computed iterate differs from the comparator $\boldsymbol{x}_t^\star$.

shift, adjusting its actions accordingly. Similarly, `OptFPRL` adapts directly, resulting in lower regret.

In Scenario 3, where cost directions change multiple times with increasing magnitudes, the limitations of FTRL become evident. The average regret fails to diminish, demonstrating its inability to handle such non-stationarity. While both OMD and `OptFPRL` respond to these variations, `OptFPRL` achieves lower regret. The advantages of `OptFPRL` are even more pronounced in Scenario 4, which involves high-frequency cost changes. That said, the observed difference between the implemented versions of OMD and `OptFPRL` is primarily due to the parameter tuning of each algorithm. The tested configurations use the theoretically optimal learning rate η_t for OMD and regularization parameter σ_t for `OptFPRL`, but different choices may lead to considerably different behavior. In contrast, the poor performance of vanilla FTRL cannot be mitigated by tuning theoretically motivated parameters–its failure is more fundamental, as discussed in the main text.

We also highlight a scenario in which `OptFPRL` performs the worst (Scenario 5): high-frequency cost switches with alternating magnitudes (large and small). This setting is deliberately designed to exploit our method's extra agility, forcing "undue" frequent adjustments. As expected, `OptFPRL` exhibits higher regret in this case, consistent with the theoretical results. Nonetheless, this tradeoff is inherent to the design and provides insights into potential extensions that balance agility and stability.

Lastly, we highlight in Scenario 6 the role of very high-quality predictions in

the performance of the *optimistic* versions of the three algorithms ([160] for OMD and [166, Sec. 7.1] for FTRL). We plot the average regret under predictions constructed as the original functions plus adversarial noise. Specifically, the adversarial noise is set as the negative of the original cost functions, with magnitude decaying quickly as $1/(0.1t)$, becoming negligible by $t \approx 100$. As noted in the paper, standard FTRL can be easily "trapped", accumulating redundant gradients and failing to track the comparators. Optimistic OGD and our OptFPRL react immediately when losses change direction.

5A

Appendix of Chapter 5

5A.1. Proof of Lemma 5.1

Define

$$\alpha_x(y) \doteq A^x \left(B \sum_{j=1}^{p} \left(M_{y-x}^{[j]} \, \boldsymbol{w}_{y-x-j} \right) + \boldsymbol{w}_{y-x} \right), x, y \in \mathbb{N}.$$

so as to write the state as

$$\boldsymbol{x}_{t+1} = \sum_{i=0}^{t} A^i \left(B \sum_{j=1}^{p} \left(M_{t-i}^{[j]} \, \boldsymbol{w}_{t-i-j} \right) + \boldsymbol{w}_{t-i} \right) = \sum_{i=0}^{t} \alpha_i(t)$$

Proof. We prove the expression by induction. For $t = 1$, (5.4) reduces to $\boldsymbol{x}_2 = \boldsymbol{w}_1$ which follows directly from the dynamic equation in (5.1) after substituting the assumptions on the initial state and actions. Then, assuming that (5.4) is true for any t, we have that

$$\boldsymbol{x}_{t+2} = A\boldsymbol{x}_{t+1} + B\boldsymbol{u}_{t+1} + \boldsymbol{w}_{t+1} = A\boldsymbol{x}_{t+1} + B \left(\sum_{j=1}^{p} M_{t+1}^{[j]} \, \boldsymbol{w}_{t+1-j} \right) + \boldsymbol{w}_{t+1}$$

$$= A \sum_{i=0}^{t} A^i \left(B \sum_{j=1}^{p} M_{t-i}^{[j]} \boldsymbol{w}_{t-i-j} + \boldsymbol{w}_{t-i} \right) + B \left(\sum_{j=1}^{p} M_{t+1}^{[j]} \boldsymbol{w}_{t+1-j} \right) + \boldsymbol{w}_{t+1}$$

$$= A \sum_{i=1}^{t+1} A^{i-1} \left(B \sum_{j=1}^{p} M_{t-i+1}^{[j]} \boldsymbol{w}_{t-i-j+1} + \boldsymbol{w}_{t-i+1} \right) + B \left(\sum_{j=1}^{p} M_{t+1}^{[j]} \boldsymbol{w}_{t+1-j} \right) + \boldsymbol{w}_{t+1}$$

$$= \sum_{i=1}^{t+1} \underbrace{A^i \left(B \sum_{j=1}^{p} M_{t-i+1}^{[j]} \boldsymbol{w}_{t-i-j+1} + \boldsymbol{w}_{t-i+1} \right)}_{\alpha_i(t+1)} + \underbrace{B \left(\sum_{j=1}^{p} M_{t+1}^{[j]} \boldsymbol{w}_{t+1-j} \right) + \boldsymbol{w}_{t+1}}_{\alpha_0(t+1)}$$

$$= \sum_{i=0}^{t+1} \alpha_i(t+1)$$

□

5A.2. Proof of Lemma 5.2

To prove Lemma 5.2, we need to make use of two known results in non-stochastic control. The first one, stated in Lemma 5A.1 characterizes the state under any linear policy. The second, stated in Lemma 5A.2, relates the state deviation between DAC and linear policies to the deviation of their actions.

Lemma 5A.1. *Assuming that $\boldsymbol{x}_1 = 0$, and parameters $\boldsymbol{w}_t$ are 0 for $t \leq 0$, the state of the system at $t+1$ upon the execution of actions $\{\boldsymbol{u}_s\}_{s=1}^{t}$ which are derived from the a linear policy parametrized by K: $\boldsymbol{u}_s = K\boldsymbol{x}_s$ can be written as:*

$$\boldsymbol{x}_{t+1} = \sum_{i=0}^{t} (A + BK)^i \boldsymbol{w}_{t-i} = \sum_{i=0}^{t} \beta_i(t), \tag{5A.1}$$

$$\text{where for any } x, y \in \mathbb{N}, \quad \beta_x(y) \doteq (A + BK)^x \ \boldsymbol{w}_{y-x}$$

Proof. We prove the expression by induction. For $t = 1$, (5A.1) reduces to $\boldsymbol{x}_2 = \boldsymbol{w}_1$ which follows directly from the dynamic equation in (5.1) after substituting the assumptions on the initial state and actions. Then, assuming that (5A.1) is true for some t, we have that

$$\begin{aligned}
\boldsymbol{x}_{t+2} &= A\boldsymbol{x}_{t+1} + B\boldsymbol{u}_{t+1} + \boldsymbol{w}_{t+1} \\
&= A \left(\sum_{i=0}^{t} (A + BK)^i \boldsymbol{w}_{t-i} \right) + B \left(K \sum_{i=0}^{t} (A + BK)^i \boldsymbol{w}_{t-i} \right) + \boldsymbol{w}_{t+1} \\
&= (A + BK) \sum_{i=0}^{t} (A + BK)^i \boldsymbol{w}_{t-i} + \boldsymbol{w}_{t+1} = \sum_{i=1}^{t+1} \underbrace{(A + BK)^i \boldsymbol{w}_{t-i+1}}_{\beta_i(t+1)} + \underbrace{\boldsymbol{w}_{t+1}}_{\beta_0(t+1)} \\
&= \sum_{i=0}^{t+1} \beta_i(t+1)
\end{aligned}$$

□

Next, we need a lemma that characterizes the state deviation between any two policies

5A

Lemma 5A.2. *let $\boldsymbol{x}_t(\pi_1)$ be the state of the system reached by following policy π_1 from the beginning of time. Analogously, let $\boldsymbol{x}_t(\pi_2)$ be the state resulting from following π_2. Let the system (A, B) be intrinsically stable (i.e., $\|A\|_{op} \leq (1-\delta)$, $\|B\| \leq 1$), and assume that the starting state and action are zero $\boldsymbol{x}_1 = \mathbf{0}, \boldsymbol{u}_1 = \mathbf{0}$. Then, the following holds:*

$$\begin{aligned}\|\boldsymbol{x}_{t+1}(\pi_1) - \boldsymbol{x}_{t+1}(\pi_2)\| &\leq \sum_{i=0}^{t} \|A^i\|_{op}\|B\| \max_{j:j\leq t} \|\boldsymbol{u}_{t-j}(\pi_1) - \boldsymbol{u}_{t-j}(\pi_2)\| \\ &\leq \frac{1}{\delta} \max_{j:j\leq t} \|\boldsymbol{u}_{t-j}(\pi_1) - \boldsymbol{u}_{t-j}(\pi_2)\|.\end{aligned}$$

In words, the deviation is fully controlled by the system stability and the maximum deviations of actions.

Proof. We proceed by induction on t to write the state $\boldsymbol{x}_{t+1}$ in terms of the previous disturbance and actions. The claim is that

$$\boldsymbol{x}_{t+1} = \sum_{i=0}^{t} A^i(B\boldsymbol{u}_{t-i} + \boldsymbol{w}_{t-i})$$

For the base case of $t = 1$, the above gives $\boldsymbol{x}_2 = \boldsymbol{w}_1$, which follows by the assumption on the initial action. Now, assuming that the statement is true for t, we have that for $t + 1$:

$$\boldsymbol{x}_{t+2} = A\boldsymbol{x}_{t+1} + B\boldsymbol{u}_{t+1} + \boldsymbol{w}_{t+1} = A\left(\sum_{i=0}^{t} A^i(B\boldsymbol{u}_{t-i} + \boldsymbol{w}_{t-i})\right) + B\boldsymbol{u}_{t+1} + \boldsymbol{w}_{t+1}$$

$$\sum_{i=1}^{t+1} A^i(B\boldsymbol{u}_{t-i+1} + \boldsymbol{w}_{t-i+1}) + B\boldsymbol{u}_{t+1} + \boldsymbol{w}_{t+1} = \sum_{i=0}^{t+1} A^i(B\boldsymbol{u}_{t-i+1} + \boldsymbol{w}_{t-i+1})$$

Subtracting the state expression reached under $\{\boldsymbol{u}_s(\pi_1)\}_{s=1}^t$, and $\{\boldsymbol{u}_s(\pi_2)\}_{s=1}^t$, the result follows by the bound on the sum of geometric series. □

Now, we show that the actions, and consequently the states, produced by any DAC policy can approximate those of a linear policy with arbitrarily small error ζ.

Proof of Lemma 5.2. let $\boldsymbol{u}_t(\pi)$ be the action produced by a stationary DAC policy π, and $\boldsymbol{u}_t(\pi^{\mathbb{L}})$ be the action produced by a linear policy $\pi^{\mathbb{L}}$. Then, by Lemma 5A.1 we have

$$\boldsymbol{u}_t(\pi^{\mathbb{L}}) = \sum_{j=0}^{t-1} K(A+BK)^j\boldsymbol{w}_{t-j-1} = \sum_{j=0}^{p-1} K(A+BK)^j\boldsymbol{w}_{t-j-1} + \sum_{j=p}^{t-1} K(A+BK)^j\boldsymbol{w}_{t-j-1}$$

$$= \underbrace{\sum_{j=0}^{p-1} M^{[j+1]} \boldsymbol{w}_{t-j-1}}_{\boldsymbol{u}_t(\pi)} + \sum_{j=p}^{t-1} K(A+BK)^j \boldsymbol{w}_{t-j-1},$$

Where we denoted the j-th polynomial in K with $M^{[j+1]}$ in the first sum. Now, note that the first sum is the DAC action. Hence,

$$\|\boldsymbol{u}_t(\pi) - \boldsymbol{u}'_t(\pi^{\mathbb{L}})\| \leq \|\sum_{j=p}^{t} K(A+BK)^j \boldsymbol{w}_{t-j}\| \overset{(\alpha)}{\leq} \sum_{j=p+1}^{t} (1-\delta)^j \kappa^{\mathbb{L}} w$$
$$\overset{(\beta)}{\leq} \kappa^{\mathbb{L}} w \int_{j=p}^{\infty} e^{-\delta j} dj = \kappa^{\mathbb{L}} \frac{w}{\delta} e^{-\delta p} \overset{(\gamma)}{\leq} \zeta$$

(α) is because K is assumed a stabilizing linear controller $\|A+BK\|_{\text{op}} \leq 1-\delta$, (β) is from $1+x \leq e^x$, and lastly (γ) by the choice of $p = 1/\delta \log\left(\kappa^{\mathbb{L}} w/\delta\zeta\right)$. This small discrepancy in the actions translates to the same one in the states (up to the stability constant) by Lemma 5A.2:

$$\|\boldsymbol{x}_{t+1}(\pi) - \boldsymbol{x}_{t+1}(\pi^{\mathbb{L}})\| \leq \frac{1}{\delta}\zeta$$

Using the fact that $c_t(\cdot,\cdot)$ is Lipschitz $c_t\big(\boldsymbol{x}_t(\pi), \boldsymbol{u}_t(\pi)\big) - c_t\big(\boldsymbol{x}_t(\pi^{\mathbb{L}}), \boldsymbol{u}_t(\pi^{\mathbb{L}})\big) = \mathcal{O}(\zeta)$ □

5A.3. The non-adaptive case (OGD with fixed learning rate)

When using OGD with fixed learning rate update as in [178], we have that

$$M_{t+1} = M_t + \eta G_t \tag{5A.2}$$

5A

The proof of Lemma 5.4 follows the same steps until

$$\nu_t \leq lz \sum_{i=0}^{t-1} (1-\delta)^i \|M_{t-i-1} - M_t\| \leq lz \sum_{i=0}^{t-1} (1-\delta)^i \sum_{\tau=t-i-1}^{t-1} \|M_{\tau+1} - M_\tau\|$$

which we now bound from (5A.2) as

$$lz \sum_{i=0}^{t-1} (1-\delta)^i \sum_{\tau=t-i-1}^{t-1} \|M_{\tau+1} - M_\tau\| \leq lz\eta \sum_{i=0}^{t-1} (1-\delta)^i (i+1) \|G_\tau\|$$
$$\leq lz\eta g \sum_{i=0}^{t-1} (1-\delta)^i (i+1) \leq \frac{lz\eta g}{\delta^2}$$

Hence we get that $v_{1:T} \leq \frac{lz\eta g}{\delta^2} T$. To prove an analogous to Theorem 5.3, we have

$$\begin{aligned}
&\sum_{t=1}^{T} \Big(c_t(\boldsymbol{x}_t, \boldsymbol{u}_t) - c_t\big(\boldsymbol{x}_t(\pi_\star), \boldsymbol{u}_t(\pi_\star)\big) \Big) \\
&\quad \leq \underbrace{\sum_{t=1}^{T} \Big(c_t\big(\boldsymbol{x}_t(\pi_t), \boldsymbol{u}_t(\pi_t)\big) - c_t\big(\boldsymbol{x}_t(\pi_\star), \boldsymbol{u}_t(\pi_\star)\big) \Big)}_{\text{Regret of OGD-based stationary policy}} + \nu_{1:T} \\
&\quad \overset{(a)}{\leq} \frac{(2\kappa_M)^2}{2\eta} + \frac{\eta}{2} g^2 T + v_{1:T} \leq \frac{2\kappa_M^2}{\eta} + \frac{\eta}{2} g^2 T + \frac{lz\eta g}{\delta^2} T
\end{aligned} \tag{5A.3}$$

where inequality (a) follows from the bound on OGD's regret [28, Thm. 2.13] (Recall that this sum is the regret of in the standard memoryless case), with $\eta_t \doteq \eta, \forall t$. optimizing η in (5A.3), we get $\eta = \frac{2\kappa_M \delta}{\sqrt{g(g\delta^2 + 2lz)}} \frac{1}{\sqrt{T}}$, substituting back we get the non-adaptive OGD bound:

$$\sum_{t=1}^{T} \Big(c_t(\boldsymbol{x}_t, \boldsymbol{u}_t) - c_t\big(\boldsymbol{x}_t(\pi_\star), \boldsymbol{u}_t(\pi_\star)\big) \Big) \leq \frac{2\kappa_M}{\delta} \sqrt{g(g\delta^2 + 2lz)} \sqrt{T}$$

5A.4. On the choice of the decision set $\mathcal{M}$

We note the different choice of the decision set $\mathcal{M}$ for DAC parametrization in [178] and some of the follow up papers, where $\mathcal{M}$ is defined as:

$$\mathcal{M} \doteq \left\{ M = \left[M^{[1]} | \dots | M^{[j]} | \dots | M^{[p]} \right] : \left\| M^{[j]} \right\| \leq \kappa_M (1-\delta)^j, \forall j \leq p \right\}.$$

I.e., the norm of the submatrices decays with parameter j. Such definition is necessary when analyzing the regret against the *optimal linear controller*, where we have seen from Lemma 5.2 that $p \propto \log T$ is necessary, hence $\|\mathcal{M}\|$ *increases with time*. Nonetheless, the exponential decay of the norm w.r.t j still ensures bounded diameter in terms of κ_M. This can be seen from e.g., [192, Lem. 20, claim (iii)]. In our case, we analyze the regret against the *optimal DAC* policy directly, and hence we use the definition:

$$\mathcal{M} \doteq \left\{ M = \left[M^{[1]} | \dots | M^{[j]} | \dots | M^{[p]} \right] \ : \ \sum_{j=1}^{p} \left\| M^{[j]} \right\| \leq \kappa_M \right\},$$

which appears also in the monograph [82, Sec. 6.2.4]. Here, p is *pre-determined* and fixed property of the class. Thus, we can use the diameter bound $\|M_1 - M_2\| \leq 2\kappa_M$ as the set $\mathcal{M}$ is fixed. We leave to future work extending the analysis provided here to analyze the regret directly against the linear class, which is feasible given the tools developed in the paper.

5A.5. On the strongly stable controller K

In this subsection, we discuss the implication of dropping the assumption $\|A\|_{\text{op}} \leq (1-\delta)$, which allowed us to have $K = 0$ as a stabilizing controller. We used the fact that $K = 0$ is a stabilizing controller at the following points

- In the cost deviation lemma (Lemma. 5.4): We used that $\boldsymbol{u}(\pi_{1,\dots,t}) = \boldsymbol{u}(\pi_t)$ if $K = 0$. In general $\boldsymbol{u}(\pi_{1,\dots,t})$ and $\boldsymbol{u}(\pi_t)$ would differ only by $\|K\|\|\boldsymbol{x_t} - \boldsymbol{x_t}(\pi_t)\|$, this is exactly the state deviation term that we bound in the above-mentioned lemma, and since $c_t(\boldsymbol{u}, \boldsymbol{x})$ is Lipschitz in both arguments, the cost deviation can still be bounded but with different constant terms.
- In Lemma 5.1, where we write out the state $\boldsymbol{x_t}$ in terms of M and $\boldsymbol{w}$. Having a non-zero K would result in the term $\|A + BK\|^i$ instead of $\|A\|^i$. In this case, we can rely on the strong stability assumption, utilized in all OCO-based control works, to bound $\|A+BK\|^i$. The strong stability assumption quantifies the classical stability assumption in control and it states that there exists a strongly stable controller K that is available as an input to our controllers. A strongly stable controller is defined next

Strong stability [181, Def. 3.1]. A linear controller K is (κ, γ)-strongly stable if there exist matrices L, H satisfying $A - BK = HLH^{-1}$, such that the following two conditions are satisfied:

- The spectral norm of L satisfies $\|L\|_{\text{op}} \leq 1 - \gamma$.
- The controller and transforming matrices are bounded, i.e.,$\|K\|_{\text{op}} \leq \kappa$ and $\|H\|_{\text{op}}\|H^{-1}\|_{\text{op}} \leq \kappa$.

Essentially, the existence of K that satisfies the strong stability assumption ensures that we can use the bound

$$\|(A + BK)^t\|_{\text{op}} \leq \|H\|_{\text{op}}\|H^{-1}\|_{\text{op}}\|L\|_{\text{op}}^t \leq \kappa(1-\delta)^t.$$

I.e., the norm $\|A + BK\|_{\text{op}}$ decays exponentially and can still be bounded in terms of geometrically decaying terms $(1-\delta)^t$.

5A

Bibliography

[1] Frank Kelly. "Charging and rate control for elastic traffic". In: *European Transactions on Telecommunications* 8 (1997), pp. 33–37.

[2] Frank Kelly, Aman Maulloo, and David Kim Hong Tan. "Rate control in communication networks: shadow prices, proportional fairness and stability". In: *Journal of the Operational Research Society* 49 (1998), pp. 237–252.

[3] Stephen P Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[4] Daniel Palomar and Mung Chiang. "A Tutorial on Decomposition Methods for Network Utility Maximization". In: *IEEE Journal on Selected Areas in Communications* 24.8 (2006), pp. 1439–1451.

[5] Mung Chiang et al. "Layering as Optimization Decomposition: A Mathematical Theory of Network Architectures". In: *Proceedings of the IEEE* 95.1 (2007), pp. 255–312.

[6] Leandros Tassiulas and Anthony Ephremides. "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks". In: *IEEE Trans. on Automatic Control* 37.12 (1992), pp. 1936–1948.

[7] Yung Yi and Mung Chiang. "Stochastic network utility maximisation—a tribute to Kelly's paper published in this journal a decade ago". In: *European Transactions on Telecommunications* 19.4 (2008), pp. 421–442.

[8] Michael J. Neely, Eytan Modiano, and Chih-Ping Li. "Fairness and Optimal Stochastic Control for Heterogeneous Networks". In: *IEEE/ACM Transactions on Networking* 16.2 (2008), pp. 396–409.

[9] Leonidas Georgiadis, Michael J. Neely, and Leandros Tassiulas. "Resource Allocation and Cross-Layer Control in Wireless Networks". In: *Found. Trends Netw.* 1.1 (2006).

[10] Michael J. Neely. "Stochastic Network Optimization with Application to Communication and Queueing Systems". In: *Synthesis Lectures on Communication Networks, Morgan & Claypool Publishers* (2010).

[11] Georgios Paschos et al. "Wireless caching: technical misconceptions and business barriers". In: *IEEE Communications Magazine* 54.8 (2016), pp. 16–22.

[12] Jose A. Ayala-Romero et al. "Experimental Evaluation of Power Consumption in Virtualized Base Stations". In: *ICC 2021 - IEEE International Conference on Communications*. 2021, pp. 1–6.

[13] A. Karapantelakis, et al. “Co-creating a cyber-physical world”. In: *Ericsson White Paper GFTL-24:000856* (July 2024), pp. 1–39.

[14] Andres Garcia-Saavedra and Xavier Costa-Pérez. “O-RAN: Disrupting the Virtualized RAN Ecosystem”. In: *IEEE Communications Standards Magazine* 5.4 (2021), pp. 96–103.

[15] Chaoyun Zhang, Paul Patras, and Hamed Haddadi. “Deep Learning in Mobile and Wireless Networking: A Survey”. In: *IEEE Communications Surveys & Tutorials* 21.3 (2019), pp. 2224–2287.

[16] Darijo Raca et al. “On Leveraging Machine and Deep Learning for Throughput Prediction in Cellular Networks: Design, Performance, and Challenges”. In: *IEEE Communications Magazine* 58.3 (2020), pp. 11–17.

[17] Yandong Shi et al. “Machine Learning for Large-Scale Optimization in 6G Wireless Networks”. In: *IEEE Communications Surveys and Tutorials* 25.4 (2023), pp. 2088–2132.

[18] Fatima Hussain et al. “Machine Learning for Resource Management in Cellular and IoT Networks: Potentials, Current Solutions, and Open Challenges”. In: *IEEE Communications Surveys and Tutorials* 22.2 (2020), pp. 1251–1275.

[19] Yang Xiao et al. “Leveraging Deep Reinforcement Learning for Traffic Engineering: A Survey”. In: *IEEE Communications Surveys and Tutorials* 23.4 (2021), pp. 2064–2097.

[20] Fengxiao Tang et al. “Survey on Machine Learning for Intelligent End-to-End Communication Toward 6G: From Network Access, Routing to Traffic Control and Streaming Adaption”. In: *IEEE Communications Surveys and Tutorials* 23.3 (2021), pp. 1578–1598.

[21] Sebastian Dörner et al. “Deep Learning Based Communication Over the Air”. In: *IEEE Journal of Selected Topics in Signal Processing* 12.1 (2018), pp. 132–143.

[22] Elad Hazan. “Introduction to Online Convex Optimization”. In: (2022).

[23] Tianyi Chen et al. “Learning and Management for Internet of Things: Accounting for Adaptivity and Scalability”. In: *Proceedings of the IEEE* 107.4 (2019), pp. 778–796.

[24] Elena Veronica Belmega et al. “Online convex optimization and no-regret learning: Algorithms, guarantees and applications”. In: *arXiv:1804.04529* (2018).

[25] Alexander Rakhlin and Karthik Sridharan. “Online learning with predictable sequences”. In: *Proc. of COLT*. 2013.

[26] Mehryar Mohri and Scott Yang. “Accelerating Online Convex Optimization via Adaptive Prediction”. In: *Proc. of AISTATS*. 2016.

[27] Pooria Joulani, András György, and Csaba Szepesvári. “A Modular Analysis of Adaptive (Non-)Convex Optimization: Optimism, Composite Objectives, and Variational Bounds”. In: *Proc. of COLT*. 2017.

[28] Francesco Orabona. “A Modern Introduction to Online Learning”. In: *CoRR abs/1912.13213* (2023).

[29] T Bektas, O Oguz, and Ouveysi I. “Designing Cost-effective Content Distribution Networks”. In: *Computers & Operations Research* 34 (2007), pp. 2436–2449.

[30] Karthikeyan Shanmugam et al. “Femtocaching: Wireless Content Delivery Through Distributed Caching Helpers”. In: *IEEE Trans. Inform. Theory* 59.12 (2013), pp. 8402–8413.

[31] Dimitris Chatzopoulos et al. “Mobile augmented reality survey: From where we are to where we go”. In: *IEEE Access* 5 (2017).

[32] Georgios S. Paschos et al. “The Role of Caching in Future Communication Systems and Networks”. In: *IEEE J. Select. Areas Commun.* 36.6 (2018), pp. 1111–1125.

[33] Giovani Gracioli et al. “A survey on cache management mechanisms for real-time embedded systems”. In: *ACM Computing Surveys (CSUR)* 48.2 (2015).

[34] Alon Cohen and Tamir Hazan. “Following the Perturbed Leader for Online Structured Learning”. In: *Proc. of the ICML*. 2015.

[35] Tim Roughgarden. *Algorithms Illuminated: Omnibus Edition*. Cambridge University Press, 2022.

[36] L. A. Belady. “A study of Replacement Algorithms for a Virtual-Storage Computer”. In: *IBM Systems Journal* 5.2 (1966), pp. 78–101.

[37] Charu Aggarwal, Joel L Wolf, and Philip S. Yu. “Caching on the World Wide Web”. In: *Trans. Knowledge Data Eng.* 11.1 (1999), pp. 94–107.

[38] J. Kangasharju, J. Roberts, and K. Ross. “Object Replication Strategies in Content Distribution Networks”. In: *Computer Communications* 25.4 (2002), pp. 376–383.

[39] Negin Golrezaei et al. “Femtocaching and Device-to-device Collaboration: A New Architecture for Wireless Video Distribution”. In: *IEEE Commun. Mag.* 51.4 (2013), pp. 142–149.

[40] Daniel D. Sleator and Robert E. Tarjan. “Amortized Efficiency of List Update and Paging Rules”. In: *Commun. ACM* 28.2 (1985), pp. 202–208.

[41] Predrag R. Jelenković and Xiaozhu Kang. “Characterizing the Miss Sequence of the LRU Cache”. In: *SIGMETRICS Perform. Eval. Rev.* 36.2 (2008), pp. 119–121.

[42] Donghee Lee et al. “On the Existence of a Spectrum of Policies That Subsumes the Least Recently Used (LRU) and Least Frequently Used (LFU) Policies”. In: *SIGMETRICS Perform. Eval. Rev.* 27.1 (1999), pp. 134–143.

[43] Georgios Paschos, George Iosifidis, and Giuseppe Caire. “Cache Optimization Models and Algorithms”. In: *FnT in Communications and Information Theory* 16.3–4 (2020), pp. 156–345.

[44] Fabrice Guillemin, Thierry Houdoin, and Stéphanie Moteau. "Volatility of YouTube content in Orange networks and consequences". In: *Proc. of ICC*. 2013.

[45] Stefano Traverso et al. "Temporal Locality in Today's Content Caching: Why It Matters and How to Model It". In: *SIGCOMM Comput. Commun. Rev.* 43.5 (2013), pp. 5–12.

[46] Felipe Olmos et al. "Catalog dynamics: Impact of content publishing and perishing on the performance of a LRU cache". In: *Proc. of ITC*. 2014.

[47] Mathieu Leconte et al. "Placing Dynamic Content in Caches with Small Population". In: *Proc. of IEEE INFOCOM*. 2016.

[48] Salah-Eddine Elayoubi and James Roberts. "Performance and Cost Effectiveness of Caching in Mobile Access Networks". In: *Proc. of ICN*. 2015.

[49] Samuel O. Somuyiwa, András György, and Deniz Gündüz. "A Reinforcement-Learning Approach to Proactive Caching in Wireless Networks". In: *IEEE J. Select. Areas Commun.* 36.6 (2018), pp. 1331–1344.

[50] Alireza Sadeghi, Fatemeh Sheikholeslami, and Georgios B. Giannakis. "Optimal and Scalable Caching for 5G Using Reinforcement Learning of Space-Time Popularities". In: *IEEE J. Select. Areas Commun.* 12.1 (2018), pp. 180–190.

[51] Yaohua Sun et al. "A Joint Learning and Game-Theoretic Approach to Multi-Dimensional Resource Management in Fog Radio Access Networks". In: *IEEE Trans. on Vehicular Technology* 72.2 (2023), pp. 2550–2563.

[52] Zhenyu Song et al. "Learning Relaxed Belady for Content Distribution Network Caching". In: *Proc. of NSDI*. 2020.

[53] Gang Yan, Jian Li, and Don Towsley. "Learning from Optimal Caching for Content Delivery". In: *Proc. of CoNEXT*. 2021.

[54] Sascha Geulen, Berthold Vöcking, and Melanie Winkler. "Regret Minimization for Online Buffering Problems Using the Weighted Majority Algorithm". In: *Proc. of COLT*. 2010.

5A

[55] Georgios S. Paschos et al. "Learning to Cache With No Regrets". In: *Proc. of IEEE INFOCOM*. 2019.

[56] Yuanyuan Li et al. "Online Caching Networks with Adversarial Guarantees". In: *Proc. ACM Meas. Anal. Comput. Syst.* 5.3 (2021).

[57] Tareq Si Salem, Giovanni Neglia, and Stratis Ioannidis. "No-Regret Caching via Online Mirror Descent". In: *Proc. of ICC*. 2021.

[58] Rajarshi Bhattacharjee, Subhankar Banerjee, and Abhishek Sinha. "Fundamental Limits on the Regret of Online Network-Caching". In: *Proc. ACM Meas. Anal. Comput. Syst.* 4.2 (2020).

[59] Debjit Paria and Abhishek Sinha. "LeadCache: Regret-Optimal Caching in Networks". In: *Proc. of NeurIPS*. 2021.

[60] Carlos Gomez-Uribe and Neil Hunt. "The Netflix Recommender System: Algorithms, Business Value, and Innovation". In: *ACM Trans. Manage. Inf. Syst.* 6.4 (2016).

[61] Xavier Amatriain. "Building Industrial-Scale Real-World Recommender Systems". In: *Proc. of RecSys*. 2012.

[62] Georgios S. Paschos, Apostolos Destounis, and George Iosifidis. "Online Convex Optimization for Caching Networks". In: *IEEE/ACM Trans. Networking* 28.2 (2020), pp. 625–638.

[63] Tareq Si Salem, Giovanni Neglia, and Stratis Ioannidis. "No-Regret Caching via Online Mirror Descent". In: *Proc. of ICC*. 2021.

[64] Tareq Si Salem, Giovanni Neglia, and Stratis Ioannidis. "No-regret caching via online mirror descent". In: *ACM Transactions on Modeling and Performance Evaluation of Computing Systems* 8.4 (2023), pp. 1–32.

[65] Debjit Paria and Abhishek Sinha. "LeadCache: Regret-Optimal Caching in Networks". In: *Proc. of NeurIPS*. 2021.

[66] Yuanyuan Li et al. "Online Caching Networks with Adversarial Guarantees". In: *Proc. ACM Meas. Anal. Comput. Syst.* 5.3 (2021).

[67] Livia Elena Chatzieleftheriou, Merkouris Karaliopoulos, and Iordanis Koutsopoulos. "Jointly Optimizing Content Caching and Recommendations in Small Cell Networks". In: *IEEE Trans. Mobile Comput.* 18.1 (2019), pp. 125–138.

[68] Theodoros Giannakas, Pavlos Sermpezis, and Thrasyvoulos Spyropoulos. "Network Friendly Recommendations: Optimizing for Long Viewing Sessions". In: *IEEE Trans. on Mobile Comput.* (2021).

[69] Yaru Fu et al. "Revenue Maximization: The Interplay Between Personalized Bundle Recommendation and Wireless Content Caching". In: *IEEE Trans. on Mobile Comput.* (2022).

[70] Thodoris Lykouris and Sergei Vassilvtiskii. "Competitive Caching with Machine Learned Advice". In: *Proc. of ICML*. 2018.

[71] Dhruv Rohatgi. "Near-Optimal Bounds for Online Caching with Machine Learned Advice". In: *Proc. of ACM-SIAM SODA*. 2020.

[72] Antonios Antoniadis et al. "Online Metric Algorithms with Untrusted Predictions". In: *Proc. of ICML*. 2020.

[73] Daan Rutten et al. "Smoothed Online Optimization with Unreliable Predictions". In: *Proc. ACM Meas. Anal. Comput. Syst.* 7.1 (2023).

[74] Lachlan Andrew et al. "A Tale of Two Metrics: Simultaneous Bounds on Competitiveness and Regret". In: *Proc. of COLT*. 2013.

[75] Naram Mhaisen, George Iosifidis, and Douglas Leith. "Online Caching With no Regret: Optimistic Learning via Recommendations". In: *IEEE Trans. Mob. Comput.* 23.5 (2024), pp. 5949–5965.

[76] Kun Chen and Longbo Huang. "Timely-Throughput Optimal Scheduling With Prediction". In: *IEEE/ACM Transactions on Networking* 26.6 (2018), pp. 2457–2470.

[77] Xi Huang et al. "Online VNF chaining and predictive scheduling: Optimality and trade-offs". In: *IEEE/ACM Transactions on Networking* 29.4 (2021), pp. 1867–1880.

[78] Zhi Zhou et al. "Predictive online server provisioning for cost-efficient iot data streaming across collaborative edges". In: *Proc. of ACM MobiHoc*. 2019.

[79] Joshua Comden et al. "Online Optimization in Cloud Resource Provisioning: Predictions, Regrets and Algorithms". In: *Proc. ACM Meas. Anal. Comput. Syst.* 1.3 (2019).

[80] Alexander Rakhlin and Karthik Sridharan. "Online Learning with Predictable Sequences". In: *Proceedings of the 26th Annual Conference on Learning Theory*. 2013, pp. 993–1019.

[81] Jacob Abernethy et al. "Online Linear Optimization via Smoothing". In: *Proceedings of The 27th Conference on Learning Theory*. 2014, pp. 807–823.

[82] Elad Hazan and Karan Singh. "Introduction to online nonstochastic control". In: *arXiv:2211.09619* (2022).

[83] Yuchao Li and Anders Rantzer. "Exact Dynamic Programming for Positive Systems With Linear Optimal Cost". In: *IEEE Trans. on Automatic Control* 69.12 (2024).

[84] David Ohlin, Emma Tegling, and Anders Rantzer. "Optimal control of linear cost networks". In: *European Journal of Control* 80 (2024), p. 101068.

[85] Elad Hazan, Sham Kakade, and Karan Singh. "The Nonstochastic Control Problem". In: *Proc. of ALT*. 2020.

[86] Genevieve E Flaspohler et al. "Online learning with optimism and delay". In: *Proc. of ICML*. 2021.

[87] Yaru Fu et al. "Revenue Maximization for Content-Oriented Wireless Caching Networks (CWCNs) With Repair and Recommendation Considerations". In: *IEEE Trans. Wireless Commun.* 20.1 (2021), pp. 284–298.

[88] Min Sheng et al. "Cooperative Content Replacement and Recommendation in Small Cell Networks". In: *IEEE Trans. Wireless Commun.* 20.3 (2021), pp. 2049–2063.

[89] Navneet Garg et al. "Online Content Popularity Prediction and Learning in Wireless Edge Caching". In: *IEEE Trans. Commun.* 68.2 (2020), pp. 1087–1100.

[90] Martin Zinkevich. "Online Convex Programming and Generalized Infinitesimal Gradient Ascent". In: *Proc. of ICML*. 2003.

[91] Ofer Dekel, Nika Haghtalab, Patrick Jaillet, et al. "Online Learning with a Hint". In: *Proc. of NeurIPS*. 2017.

5A

[92] Aditya Bhaskara et al. “Online Learning with Imperfect Hints”. In: *Proc. of ICML*. 2020.

[93] Alexander Rakhlin and Karthik Sridharan. “Optimization, Learning, and Games with Predictable Sequences”. In: *Proc. of NeurIPS*. 2013.

[94] Shai Shalev-Shwartz and Yoram Singer. “A Primal-Dual Perspective of Online Learning Algorithms”. In: *Mach. Learn.* 69.2-3 (2007), pp. 115–142.

[95] Alireza Sadeghi et al. “Reinforcement Learning for Adaptive Caching With Dynamic Storage Pricing”. In: *IEEE J. Select. Areas Commun.* 37.10 (2019), pp. 2267–2281.

[96] Jeongho Kwak, Georgios Paschos, and George Iosifidis. “Dynamic Cache Rental and Content Caching in Elastic Wireless CDNs”. In: *Proc. of WiOpt*. 2018.

[97] Erik Nygren, Ramesh K. Sitaraman, and Jennifer Sun. “The Akamai Network: A Platform for High-Performance Internet Applications”. In: *SIGOPS Oper. Syst. Rev.* 44.3 (2010), pp. 2–19.

[98] Tianyi Chen, Qing Ling, and Georgios B. Giannakis. “An Online Convex Optimization Approach to Proactive Network Resource Allocation”. In: *IEEE Trans. Signal Processing* 65.24 (2017), pp. 6350–6364.

[99] Nikolaos Liakopoulos et al. “Cautious Regret Minimization: Online Optimization with Long-Term Budget Constraints”. In: *Proc. of ICML*. 2019.

[100] Victor Valls et al. “Online Convex Optimization with Perturbed Constraints: Optimal Rates against Stronger Benchmarks”. In: *Proc. of AISTATS*. 2020.

[101] Xinlei Yi et al. “Distributed Online Convex Optimization With Time-Varying Coupled Inequality Constraints”. In: *IEEE Trans. Signal Processing* 68 (2020), pp. 731–746.

[102] Zhihao Shen et al. “DeepAPP: A Deep Reinforcement Learning Framework for Mobile Application Usage Prediction”. In: *IEEE Trans. on Mobile Comput.* (2021).

[103] J. Ben Schafer, Joseph A. Konstan, and John Riedl. “Meta-Recommendation Systems: User-Controlled Integration of Diverse Recommendations”. In: *Proc. of CIKM*. 2002.

[104] Michael Zink et al. “Characteristics of YouTube Network Traffic at a Campus Network - Measurements, Models, and Implications”. In: *Comput. Netw.* 53.4 (2009), pp. 501–514.

[105] F. Maxwell Harper and Joseph A. Konstan. “The MovieLens Datasets: History and Context”. In: *ACM Trans. Interact. Intell. Syst.* 5.4 (2015).

[106] Georgios S. Paschos and Ejder Bastug and Ingmar Land and Giuseppe Caire, and Merouane Debbah. “Wireless Caching: Technical Misconceptions and Business Barriers”. In: *IEEE Commun. Mag.* 54.8 (2016), pp. 16–22.

[107] Xi Huang et al. “Online User-AP Association with Predictive Scheduling in Wireless Caching Networks”. In: *IEEE Trans. Mobile Comput.* (2020).

[108] Anastasios Giovanidis and Apostolos Avranas. “Spatial Multi-LRU Caching for Wireless Networks with Coverage Overlaps”. In: *SIGMETRICS Perform. Eval. Rev.* 44.1 (2016), pp. 403–405.

[109] Emilio Leonardi and Giovanni Neglia. “Implicit Coordination of Caches in Small Cell Networks Under Unknown Popularity Profiles”. In: *IEEE J. Select. Areas Commun.* 36.6 (2018), pp. 1276–1285.

[110] Dimitra Tsigkari and Thrasyvoulos Spyropoulos. “User-centric Optimization of Caching and Recommendations in Edge Cache Networks”. In: *Proc. of WoWMoM*. 2020.

[111] Lijun Zhang et al. “Dynamic Regret of Strongly Adaptive Methods”. In: *Proc. of ICML*. 2018.

[112] H. Brendan McMahan. “A Survey of Algorithms and Analysis for Adaptive Online Learning”. In: *J. Mach. Learn. Res.* 18.1 (2017), pp. 3117–3166.

[113] Amir Beck. “First-Order Methods in Optimization”. In: (2017).

[114] Peter Auer, Nicolò Cesa-Bianchi, and Claudio Gentile. “Adaptive and Self-Confident On-Line Learning Algorithms”. In: *Journal of Computer and System Sciences* 64.1 (2002), pp. 48–75.

[115] Georgios S. Paschos, Apostolos Destounis, and George Iosifidis. “Online Convex Optimization for Caching Networks”. In: *IEEE/ACM Trans. Networking* 28.2 (2020), pp. 625–638.

[116] *Amazon Elastic CDN Service - ElastiCache*. URL: `https://aws.amazon.com/elasticache/`.

[117] Juniper Networks. *The Elastic CDN Solution*. Solution Brief, Dec. 2014. URL: `https://www.juniper.net/assets/kr/kr/local/pdf/solutionbriefs/3510532-en.pdf`.

[118] Shie Mannor, John N. Tsitsiklis, and Jia Yuan Yu. “Online Learning with Sample Path Constraints”. In: *J. Mach. Learn. Res.* 10 (2009), pp. 569–590.

[119] Daron Anderson and Douglas J. Leith. *Learning The Best Expert Efficiently*. URL: `https://arxiv.org/abs/1911.04307`.

[120] Yang Zhang et al. “How to Retrain Recommender System? A Sequential Meta-Learning Method”. In: *Proc. of SIGIR*. 2020.

[121] Naram Mhaisen. *online-caching*. Apr. 12, 2022. URL: `https://github.com/Naram-m/online-caching`.

[122] Weiran Wang and Canyi Lu. *Projection onto the capped simplex*. 2015. URL: `https://arxiv.org/abs/1503.01002`.

[123] Naram Mhaisen, George Iosifidis, and Douglas Leith. “Online Caching with Optimistic Learning”. In: *Proc. of IFIP Networking*. 2022.

[124] M. A. Maddah-Ali, and U. Niesen. “Fundamental Limits of Caching”. In: *IEEE Trans. Inf. Theory* 60.5 (2014), pp. 2856–2867.

5A

[125] Alexander Rakhlin and Karthik Sridharan. "Optimization, Learning, and Games with Predictable Sequences". In: *Proc. of NeurIPS*. 2013.

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

[127] Jarosław Byrka et al. "An Improved Approximation for K-Median and Positive Correlation in Budgeted Optimization". In: *ACM Trans. Algorithms* 13.2 (2017).

[128] George B. Dantzig. "Discrete-Variable Extremum Problems". In: *Operations Research* 5.2 (1957), pp. 266–277.

[129] Jacob Abernethy et al. "Online Linear Optimization via Smoothing". In: *Proc. of COLT*. 2014.

[130] Sarah Sachs et al. "Between Stochastic and Adversarial Online Convex Optimization: Improved Regret Bounds via Smoothness". In: *Proc. of NeurIPS*. 2022.

[131] Steven De Rooij et al. "Follow the Leader If You Can, Hedge If You Must". In: *J. Mach. Learn. Res.* 15.1 (2014), pp. 1281–1316.

[132] Daron Anderson, George Iosifidis, and Douglas Leith. "Lazy Lagrangians for Optimistic Learning With Budget Constraints". In: *IEEE/ACM Trans. on Networking* 31.5 (2023), pp. 1935–1949.

[133] Arun Sai Suggala and Praneeth Netrapalli. "Online Non-Convex Learning: Following the Perturbed Leader is Optimal". In: *Proc. of ALT*. 2020.

[134] Arun Sai Suggala and Praneeth Netrapalli. "Follow the Perturbed Leader: Optimism and Fast Parallel Algorithms for Smooth Minimax Games". In: *Proc. of NeurIPS*. 2020.

[135] Noga Alon and Joel H Spencer. *The probabilistic method*. John Wiley & Sons, 2016.

[136] William G. Madow. "On the Theory of Systematic Sampling". In: *The Annals of Mathematical Statistics* 20.3 (1949), pp. 333–354.

[137] Thomas H Cormen et al. *Introduction to algorithms*. MIT press, 2022.

[138] Dimitri P Bertsekas. "Stochastic optimization problems with nondifferentiable cost functionals". In: *Journal of Optimization Theory and Applications* 12.2 (1973), pp. 218–231.

[139] Marcus Hutter and Jan Poland. "Adaptive Online Prediction by Following the Perturbed Leader". In: *Journal of Machine Learning Research* 6.22 (2005), pp. 639–660.

[140] Silvano Martello and Paolo Toth. *Knapsack Problems: Algorithms and Computer Implementations*. J. Willey & Sons, 1990.

[141] Adam Kalai and Santosh Vempala. "Efficient algorithms for online decision problems". In: *Journal of Computer and System Sciences* 71.3 (2005), pp. 291–307.

[142] Vijay V Vazirani. *Approximation algorithms*. Vol. 1. Springer, 2001.

[143] Robert Gramacy et al. "Adaptive Caching by Refetching". In: *Proc. of NIPS*. 2002.

[144] Liana Rodriguez et al. "Learning Cache Replacement with Cacheus". In: *Proc. of USENIX Conferecne on File and Storage Technologies*. 2021.

[145] Naram Mhaisen. *Opt-FPRL*. 2023. URL: `https://github.com/Naram-m/discrete-online-learning`.

[146] Geoffrey J Gordon, Amy Greenwald, and Casey Marks. "No-regret learning in convex games". In: *Proceedings of the 25th international conference on Machine learning*. 2008, pp. 360–367.

[147] Ativ Joshi and Abhishek Sinha. "Universal Caching". In: *arXiv preprint arXiv:2205.04860* (2022).

[148] Konstantinos Poularakis, George Iosifidis, and Leandros Tassiulas. "Approximation Algorithms for Mobile Data Caching in Small Cell Networks". In: *IEEE Trans. Commun.* 62.10 (2014), pp. 3665–3677.

[149] Samrat Mukhopadhyay, Sourav Sahoo, and Abhishek Sinha. "k-experts-Online Policies and Fundamental Limits". In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2022, pp. 342–365.

[150] Naram Mhaisen et al. "Optimistic No-regret Algorithms for Discrete Caching". In: *Proc. ACM Meas. Anal. Comput. Syst.* 6.3 (2022), pp. 1–28.

[151] Chung-En Tsai, Ying-Ting Lin, and Yen-Huan Li. "Data-dependent bounds for online portfolio selection without Lipschitzness and smoothness". In: *Proc. of NeurIPS*. Vol. 36. 2024.

[152] Fatih Aslan et al. "Fair Resource Allocation in Virtualized O-RAN Platforms". In: *Proc. ACM Meas. Anal. Comput. Syst.* 8.1 (2024).

[153] Kwangjun Ahn et al. "Understanding Adam optimizer via online learning of updates: Adam is FTRL in disguise". In: *Proc. of ICML*. 2024.

[154] Tareq Si-Salem et al. "Online Submodular Maximization via Online Convex Optimization". In: *Proc. of AAAI*. 2024.

[155] Yong Bai et al. "Adapting to online label shift with provable guarantees". In: *Proc. of NeurIPS*. 2022.

[156] Shai Shalev-Shwartz. "Online Learning and Online Convex Optimization". In: *FnT in Machine Learning* 4.2 (2012), pp. 107–194.

[157] john Duchi, Elad Hazan, and Yoram Singer. "Adaptive subgradient methods for online learning and stochastic optimization". In: *JMLR* 12.61 (2011), pp. 2121–2159.

[158] Chao-Kai Chiang et al. "Online optimization with gradual variations". In: *Proc. of COLT*. 2012.

[159] Naram Mhaisen and George Iosifidis. "Optimistic Online Non-stochastic Control via FTRL". In: *Proc. of IEEE CDC*. 2024.

5A

[160] Ali Jadbabaie et al. “Online optimization: Competing with dynamic comparators”. In: *Proc. of AISTATS*. 2015.

[161] Pedro Zattoni Scroccaro, Arman Sharifi Kolarijani, and Peyman Mohajerin Esfahani. “Adaptive Composite Online Optimization: Predictions in Static and Dynamic Environments”. In: *IEEE Trans. on Automatic Control* 68.5 (2023), pp. 2906–2921.

[162] Andrew Jacobsen and Ashok Cutkosky. “Parameter-free mirror descent”. In: *PRoc. of COLT*. 2022.

[163] Sijia Chen et al. “Optimistic online mirror descent for bridging stochastic and adversarial online convex optimization”. In: *JMLR* 25.178 (2024), pp. 1–62.

[164] Francesco Orabona and Dávid Pál. “Scale-free online learning”. In: *Theoretical Computer Science* 716 (2018), pp. 50–69.

[165] H. Brendan McMahan and Matthew Streeter. “Adaptive bound optimization for online convex optimization”. In: *Proc. of COLT*. 2010.

[166] Pooria Joulani, András György, and Csaba Szepesvári. “A Modular Analysis of Adaptive (non-)Convex Optimization: Optimism, Composite objectives, Variance Reduction, and Variational Bounds”. In: *Theor. Comput. Sci.* 808 (2020), pp. 108–138.

[167] Lijun Zhang, Shiyin Lu, and Zhi-Hua Zhou. “Adaptive online learning in dynamic environments”. In: *Proc. of NeurIPS*. 2018.

[168] Peng Zhao et al. “Dynamic regret of convex and smooth functions”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 12510–12520.

[169] Peng Zhao et al. “Adaptivity and non-stationarity: Problem-dependent dynamic regret for online convex optimization”. In: *JMLR* 25.98 (2024), pp. 1–52.

[170] Huang Fang et al. “Online mirror descent and dual averaging: keeping pace in the dynamic case”. In: *JMLR* 23.121 (2022), pp. 1–38.

[171] Zhiyu Zhang, David Bombara, and Heng Yang. “Discounted Adaptive Online Learning: Towards Better Regularization”. In: *Proc. of ICML*. 2024.

[172] Aditya Bhaskara and Kamesh Munagala. “Competing against Adaptive Strategies in Online Learning via Hints”. In: *Proc. of AISTATS*. 2023.

[173] Andrew Jacobsen and Francesco Orabona. “An Equivalence Between Static and Dynamic Regret Minimization”. In: *Proc. of NeurIPS*. 2024.

[174] Omar Besbes, Yonatan Gur, and Assaf Zeevi. “Non-stationary stochastic optimization”. In: *Operations research* 63.5 (2015), pp. 1227–1244.

[175] Nicolo Campolongo and Francesco Orabona. “A closer look at temporal variability in dynamic online learning”. In: *arXiv:2102.07666* (2021).

[176] Daniel Suo et al. “Machine learning for mechanical ventilation control”. In: *arXiv:2102.06779* (2021).

[177] Russell Lee et al. “Online peak-aware energy scheduling with untrusted advice”. In: *ACM SIGENERGY Energy Informatics Review* 1.1 (2021), pp. 59–77.

[178] Naman Agarwal et al. “Online Control with Adversarial Disturbances”. In: *Proc. of ICML*. 2019.

[179] Oren Anava, Elad Hazan, and Shie Mannor. “Online Learning for Adversaries with Memory: Price of Past Mistakes”. In: *Proc. of NeurIPS*. 2015.

[180] Shai Shalev-Shwartz. “Online Learning and Online Convex Optimization”. In: *Foundations and Trends in Machine Learning* 4.2 (2012). ISSN: 1935-8237.

[181] Alon Cohen et al. “Online linear quadratic control”. In: *Proc. of ICML*. 2018.

[182] Zhiyu Zhang, Ashok Cutkosky, and Ioannis Paschalidis. “Adversarial tracking control via strongly adaptive online learning with memory”. In: *Proc. of AISTATS*. 2022.

[183] Anders Rantzer. “Explicit Solution to Bellman Equation for Positive Systems with Linear Cost”. In: *Proc. of CDC*. 2022.

[184] Max Simchowitz. “Making non-stochastic control (almost) as easy as stochastic”. In: *Proc. of NeurIPS*. 2020.

[185] Naman Agarwal, Elad Hazan, and Karan Singh. “Logarithmic Regret for Online Control”. In: *Proc. of NeurIPS*. 2019.

[186] Dylan Foster and Max Simchowitz. “Logarithmic Regret for Adversarial Online Control”. In: *Proc. of ICML*. 2020.

[187] Yingying Li, Subhro Das, and Na Li. “Online Optimal Control with Affine Constraints”. In: *Proc. of the AAAI*. 2021.

[188] Xin Liu, Zixian Yang, and Lei Ying. “Online Nonstochastic Control with Adversarial and Static Constraints”. In: *arXiv:2302.02426* (2023).

[189] Paula Gradu, John Hallman, and Elad Hazan. “Non-Stochastic Control with Bandit Feedback”. In: *Proc. of NeurIPS*. 2020.

[190] Paula Gradu, Elad Hazan, and Edgar Minasyan. “Adaptive Regret for Control of Time-Varying Dynamics”. In: *Proc. of L4DC*. 2023.

[191] Hongyu Zhou, Zirui Xu, and Vasileios Tzoumas. “Efficient Online Learning with Memory via Frank-Wolfe Optimization: Algorithms with Bounded Dynamic Regret and Applications to Control”. In: *Proc. of CDC*. 2023.

[192] Peng Zhao, Yu-Xiang Wang, and Zhi-Hua Zhou. “Non-stationary Online Learning with Memory and Non-stochastic Control”. In: *Proc. of AISTATS*. 2022.

[193] Guanya Shi et al. “Online optimization with memory and competitive control”. In: *Proc. of NeurIPS*. 2020.

[194] Gautam Goel and Babak Hassibi. “Competitive control”. In: *IEEE Trans. Autom. Control* 68.9 (2023), pp. 5162–5173.

5A

[195] Max Simchowitz, Karan Singh, and Elad Hazan. "Improper Learning for Non-Stochastic Control". In: *Proc. of COLT*. 2020.

[196] Gautam Goel et al. "Best of Both Worlds in Online Control: Competitive Ratio and Policy Regret". In: *Proc. of L4DC*. 2023.

[197] Tiancheng Jin et al. "No-Regret Online Reinforcement Learning with Adversarial Losses and Transitions". In: *arXiv:2305.17380* (2023).

[198] Nicolo Cesa-Bianchi, Ofer Dekel, and Ohad Shamir. "Online learning with switching costs and other adaptive adversaries". In: *Proc. of NeurIPS*. 2013.

[199] Aren Karapetyan, Andrea Iannelli, and John Lygeros. "On the regret of H∞ control". In: *Proc. of IEEE CDC*. 2022.

[200] Eyal Gofer. "Higher-order regret bounds with switching costs". In: *Proc. of COLT*. 2014.

[201] Auer et al. "Adaptive and Self-Confident On-Line Learning Algorithms". In: *Journal of Computer and System Sciences* 64.1 (2002), pp. 48–75. ISSN: 0022-0000.

[202] *Adaptive-NSC*. 2024. URL: `https://github.com/Naram-m/NSC`.

[203] Niangjun Chen et al. "Online convex optimization using predictions". In: *Proc. of SIGMETRICS*. 2015.

[204] Niangjun Chen et al. "Using predictions in online optimization: Looking forward with an eye on the past". In: *SIGMETRICS Perform. Eval. Rev.* 44.1 (2016).

[205] Chenkai Yu et al. "The Power of Predictions in Online Control". In: *Proc. of NeurIPS*. 2020.

[206] Yingying Li, Xin Chen, and Na Li. "Online Optimal Control with Linear Dynamics and Predictions: Algorithms and Regret Analysis". In: *Proc. of NeurIPS*. 2019.

[207] Runyu Zhang, Yingying Li, and Na Li. "On the regret analysis of online LQR control with predictions". In: *Proc. of ACC*. 2021.

[208] Chenkai Yu et al. "Competitive control with delayed imperfect information". In: *Proc. of ACC*. 2022.

[209] Tongxin Li et al. "Robustness and Consistency in Linear Quadratic Control with Untrusted Predictions". In: *Proc. ACM Meas. Anal. Comput. Syst.* 6.1 (2022).

[210] Orin Levy and Yishay Mansour. "Optimism in face of a context: Regret guarantees for stochastic contextual MDP". In: *Proc. of AAAI*. 2023.

[211] Alberto Bemporad and Manfred Morari. "Robust model predictive control: A survey". In: *Robustness in identification and control*. Springer, 2007.

[212] Geir E Dullerud and Fernando Paganini. *A course in robust control theory: a convex approach*. Vol. 36. Springer Science & Business Media, 2013.

[213] Yiheng Lin et al. “Perturbation-based regret analysis of predictive control in linear time varying systems”. In: *Proc. of NeurIPS*. 2021.

[214] Yiheng Lin et al. “Bounded-Regret MPC via Perturbation Analysis: Prediction Error, Constraints, and Nonlinearity”. In: *Proc. of NeurIPS*. 2022.

[215] Tareq Si-Salem et al. “Online Submodular Maximization via Online Convex Optimization”. In: *Proc. of AAAI*. 2024.

[216] Elad Hazan. “Introduction to Online Convex Optimization”. In: *arXiv:1909.05207* (2019).

[217] *Optimistic-NSC*. 2024. URL: `https://github.com/Naram-m/Optimistic-NSC`.

[218] Aditya Bhaskara et al. “Online Learning with Imperfect Hints”. In: *Proceedings of the 37th International Conference on Machine Learning*. 2020, pp. 822–831.

[219] Dacheng Wen, Yupeng Li, and Francis CM Lau. “Augment online linear optimization with arbitrarily bad machine-learned predictions”. In: *Proc. of IEEE INFOCOM*. 2024.

[220] Dacheng Wen et al. “Robust Decentralized Online Optimization Against Malicious Agents”. In: *Proc. of IEEE International Conference on Distributed Computing Systems (ICDCS)*. 2024.

[221] Yibo Wang et al. “Online composite optimization between stochastic and adversarial environments”. In: *Proc. of NeurIPS*. 2024.

[222] L. Zhang, W. Jiang, S. Lu, and T. Yang. “Revisiting Smoothed Online Learning”. In: *Proc. of NeurIPS*. 2021.

[223] Lijun Zhang, Shiyin Lu, and Tianbao Yang. “Minimizing dynamic regret and adaptive regret simultaneously”. In: *Proc. of AISTATS*. 2020.

[224] Mohammad Abu Alsheikh et al. “Markov decision processes with applications in wireless sensor networks: A survey”. In: *IEEE Communications Surveys & Tutorials* 17.3 (2015), pp. 1239–1267.

[225] Ning Yang et al. “Beyond the edge: An advanced exploration of reinforcement learning for mobile edge computing, its applications, and future research trajectories”. In: *IEEE Communications Surveys & Tutorials* (2024).

[226] Yan Dai, Haipeng Luo, and Liyu Chen. “Follow-the-perturbed-leader for adversarial markov decision processes with bandit feedback”. In: *Proc. of NeurIPS*. 2022.

[227] Genevieve E Flaspohler et al. “Online Learning with Optimism and Delay”. In: *Proceedings of the 38th International Conference on Machine Learning*. 2021.

[228] Naram Mhaisen. *Opt-FPRL*. 2025. URL: `https://github.com/Naram-m/OptFPRL`.