LEARNING FROM INTERACTIONS VIA ONLINE DECISION-MAKING AND NETWORK SCIENCE

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Raunak Kumar

August 2024

LEARNING FROM INTERACTIONS VIA ONLINE DECISION-MAKING AND NETWORK SCIENCE

Raunak Kumar, Ph.D.

Cornell University 2024

Interactions between a learner and an environment arise in a variety of domains, ranging from online recommendations (e.g., Spotify) to control of physical dynamical systems (e.g., temperature regulation in a datacenter). In this dissertation we study such problems through two different perspectives: online decision-making and network science.

In Part I we study how a learner should make decisions *while* receiving feedback from interactions online. We generalize classic models of online decision-making with a focus on settings where the learner's present outcome depends on all of their past decisions. First, we introduce a natural generalization of the stochastic bandits with knapsacks problem [Badanidiyuru et al., 2018] that allows non-monotonic resource utilization. We develop algorithms with constant and logarithmic regret against a linear programming relaxation of the problem when the outcome distributions are known and unknown to the decision-maker respectively. Next, we introduce a generalization of the online convex optimization problem [Zinkevich, 2003] that captures long-term dependence on past decisions. We prove matching upper and lower bounds on regret, including the first non-trivial lower bound for online convex optimization with finite memory [Anava et al., 2015]. We use our framework to derive regret bounds, and to improve and simplify existing regret bound derivations, for a variety of online learning problems such as online linear control and online performative prediction.

In Part II we study how a learner can analyze the interactions *after* they have occurred. We consider the projection of past interactions onto a weighted graph and generalize classic network science tools for unweighted graphs to the weighted case. In particular, we generalize a classic network science tool, triangle (3-clique) counting and enumeration [Avron, 2010, Eden et al., 2017, Kolda et al., 2013, Berry et al., 2015, Stefani et al., 2017a], to the weighted case. Given a weighted graph and an integer k, we develop deterministic and randomized sampling algorithms that retrieve the top-k weighted triangles. These algorithms perform well across a broad range of large weighted graphs and provide orders of magnitude speedup over an intelligent brute-force enumeration approach.

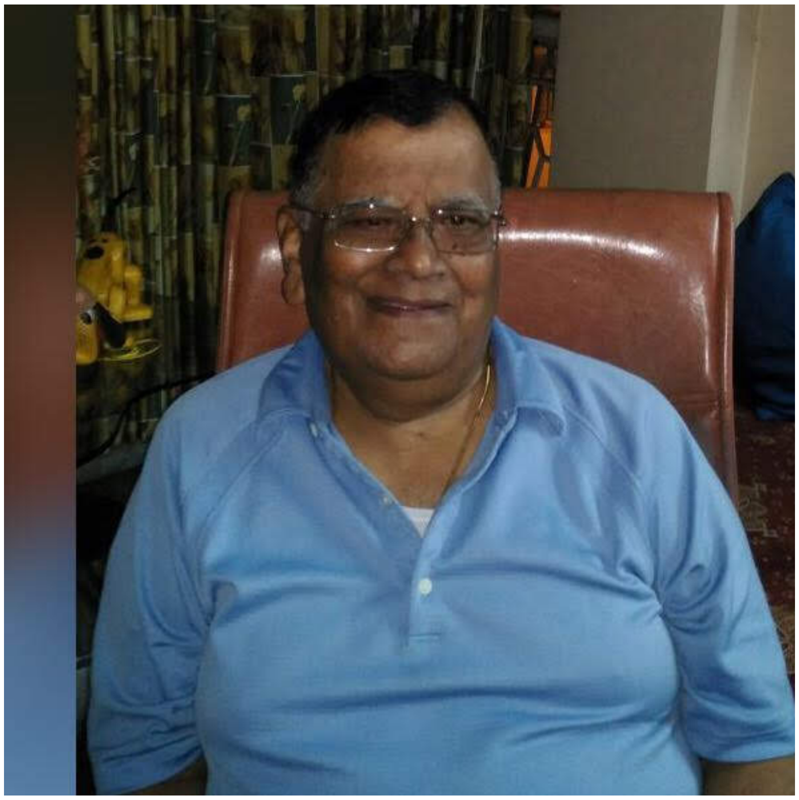
Advisors: Robert D. Kleinberg and Sarah Dean.

Additional Special Committee Members: Karthik Sridharan and Éva Tardos.

BIOGRAPHICAL SKETCH

Raunak Kumar was born in Bangalore, India. Although originally from the state of Bihar, he primarily grew up in Kolkata in the state of West Bengal. He received his undergraduate degree in computer science in 2018 from The University of British Columbia, which is located in the beautiful Canadian city of Vancouver. During his undergraduate studies, he took part in many programming competitions and was part of UBC's ACM-ICPC team. He had the pleasure of doing undergraduate research with Dr. Mark Schmidt on optimization for machine learning. Inspired by this experience, he joined Cornell University in 2018 to pursue a Ph.D. in computer science. At Cornell, he has worked broadly on online decision-making and network science. During his undergraduate studies, he interned as a software engineer at Google. During his Ph.D., he has interned as an applied scientist in the MSAI team at Microsoft over four summers, working on graphs, machine learning, and copilots. After graduating from Cornell, he will join the same team at Microsoft full-time as an applied scientist.

This dissertation is dedicated to my maternal grandfather, Ramesh Chandra Jha. He raised me like a son and loved me immensely. He did not get the opportunity to see my Ph.D. journey, but I know he would have been very proud.



ACKNOWLEDGEMENTS

First and foremost, I would like to thank my advisors, Robert (Bobby) Kleinberg and Sarah Dean. I am deeply grateful to Bobby and Sarah for being stellar role models as researchers, mentors and human beings, and for believing in me even when I didn't.

Bobby has been an incredible advisor. He is easily one of the smartest and nicest people that I have met. I feel lucky to have had Bobby as my advisor not just because of his invaluable research and technical advice, but also because of his support when things were tough. I started working with Bobby in my third year, after changing advisors and research areas for the umpteenth time. That year was very difficult for me due to a variety of reasons. I would have likely quit the Ph.D. program if not for Bobby. He was unbelievably supportive and motivating. I have learnt from him not just how to do multi-armed bandits research, but also how to be precise in research and communication, and so much more. I will be eternally grateful to him for taking a bet on me when things were tough, and for giving me the space and confidence to develop as a researcher. Without him, I would not be writing this dissertation. The principles that he abides by in research and in life are admirable, and I truly look up to him.

I am grateful to have met Sarah when I was in search of a new research project. She helped me a develop a new perspective on my research through the lens of control theory and has taught me everything that I know about control. She was always willing to listen through my ideas for problem statements and proofs, and provide feedback. Collaborating with Sarah led to a large part of this dissertation, including inspiration for the thesis and the title. I thoroughly enjoyed Sarah's group meetings, which provided an open and fun environment for us to discuss research and many other things.

I would like to thank my committee members, Karthik Sridharan and Éva Tardos, for their thought-provoking questions and support. The research in this dissertation would have remained a dream if not for my collaborators, Bobby, Sarah, Austin Benson, Moses Charikar, and Paul Liu. They taught me a ton about online learning, control, and network science.

My excitement for machine learning, algorithms, and theoretical computer science is in large part due to Mark Schmidt (my undergraduate advisor), Will Evans, and Nick Harvey. I learnt so much from taking and TAing their classes, and doing research with Mark. I would like to thank them for nurturing my interest in these areas. Without that, I would not have had the motivation and the foundation I needed to pursue a PhD.

The UBC ACM-ICPC community was a home for me during undergrad, with lots of friendship, programming contests, and, of course, pizza. I would like to thank Paul Liu in particular. Paul taught me (and numerous others) competitive programming, and did so in the most encouraging and inspiring way possible. This played a huge role in developing my problem solving skills and in me continuing to pursue algorithms and competitive programming. I deeply appreciate Paul's support.

I would like to acknowledge my family for their love, encouragement and support, and for being my biggest cheerleaders. My girlfriend, Estrella, for encouraging me to be myself and pursue what makes me happy; for being excited at my successes, no matter how small; for making me feel more comfortable with uncertainty; for comforting me with her love and support; for being my best friend; for being an amazing partner who I want to spend the rest of my life with. My maternal grandparents, Ramesh and Veena, for raising me like a son; for pampering me with a tremendous amount of love. My mother, Seema, for loving me; for making sacrifices so that I could have a good life. My uncle, Rajesh, for being my biggest supporter in so many different ways; for being a father figure; for being a loving and rational sounding board no matter the circumstance; for being a role model I look up to. My aunt, Sudha, for being so welcoming; for being a tremendous source of support in various aspects of my life. My cousins, Rohan and Rishi, for being my brothers; for being a source of support and comfort no matter the circumstance; for teaching me numerous things, including the rules of American Football; for contributing to so many fun memories. Jordan and Priya, for encouraging me with your excitement at my successes; for being a source of joy and laughter. Estrella's mom, dad and Saph for welcoming me into their family with so much warmth and love; for showing me the sights and tastes of Mexico.

I would like to thank my friends for their wonderful companionship and the cherished memories. I found a deep sense of community thanks to my friends in Ithaca through dinner parties that I will always remember fondly, playing guitar (thanks, Kevin!), playing PS5 (thanks, Abhishek!), playing soccer (thanks, CISFC!) and so many other fun activities and conversations. I am very thankful to: Kate, Katie, Katy, Cathy, Abhishek, Kevin, Megan, Sloan, Varsha, Oli, Greg, Shreyas, Kiran, Giannis, Nitika, Anshuman, Aaron, Vaishnavi, Marios, Jesse, Rachit, Kwang, Jyun-Jie, Michela, Ali, Aditya, Mohit, Yahya, Jerry, Tegan, Julia, Daniel, Rohan, Top, Suraaj, and Madhav. In my final year, I met a new set of wondeful people from the first-year cohort: Aditya, Albert, Arjun, August, Chido, Daniel, Erica, Kei, Linda, Nico, Sophie, Spencer, and Vaibhav. It's been a pleasure to overlap with them. I am also thankful to have an amazing set of friends from my time at UBC: Carlos, Kyle, Vaastav, Radu, David Chong, Daniel, Syed, Umair, Vincent, David Zheng, Paul, Chris; and from my childhood: Avinash, Gaurav, Pranay, and Rishabh. To anyone I have missed, know that I have fond memories of our friendship.

I would also like to thank my therapist, Anna, for helping me improve my mental and emotional health when I needed it the most, and my swimming instructor, Susie, for teaching me how to swim and for so many fun swim workouts, which have been a source of immense fun and relaxation.

I spent four wonderful summers interning as an applied scientist at Microsoft. I was lucky to have wonderful mentors: Jennifer Neville, Vipul Agarwal, Omar Zia Khan, Cassiano Becker, Nabiha Asghar, Kunho Kim, and Karthik Tangirala. I am also grateful to numerous other people in the MSAI team for creating an open and fun work environment.

Finally, I would like to acknowledge funding from the Natural Sciences and Engineering Research Council of Canada (NSERC). Support from the NSERC Postgraduate Scholarships-Doctoral Fellowship (545847-2020) provided me the freedom to pursue my interests during a challenging time of my PhD.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

OVERVIEW OF THE DISSERTATION

Interactions between a learner and an environment arise in a variety of domains, ranging from online recommendations to control of physical dynamical systems. For example, consider the following interaction loop in an online recommendations setting: (i) the audio streaming platform Spotify (learner) recommends content (songs, podcasts, and audiobooks) to its users (environment); (ii) it receives feedback from the users, e.g., whether they played the recommended content, how long they played it for, whether they added it to a playlist, etc.; (iii) the process repeats. For an example in the setting of controlling physical dynamical systems, consider the following interaction loop for temperature regulation in a data center [Lazic et al., 2018]: (i) the data center controller (learner) chooses actions (fan speed, valve opening, etc.); (ii) the controller receives feedback from the data center (environment) in the form of measurements such as temperature and differential air pressure; (iii) the process repeats. Given the prevalence of digital systems in our lives, it is not surprising to see such interaction loops arise in numerous other problems, such as dynamic pricing [Besbes and Zeevi, 2009, Babaioff et al., 2012], pay-per-click ad allocation [Slivkins, 2011, 2014], and variable-speed wind turbines in wind energy power production [Boukhezzar and Siguerdidjane, 2010].

In this dissertation we study such problems through two different perspectives: online decision-making and network science. In Part I we study how a learner should make decisions *while* interacting with an environment. We generalize classic models of online decision-making with a focus on settings where the learner's present outcome depends on all of their past decisions. In Part II we study how a learner can analyze the interactions *after* they have occurred. We consider the projection of past interactions onto a weighted graph and generalize classic network science tools for unweighted graphs to the weighted case.

1.1 Learning with Interactions via Online Decision-making

Online decision-making problems consist of repeated interactions between a learner and an environment over T rounds. In each round, the learner chooses a decision, suffers some loss, and observes some feedback from the environment. Clearly, the learner would like to choose a decision that minimizes the total loss over T rounds. One of the main challenges of online decision-making is that the learner does not know the environment ahead of time. So, they must choose decisions online based on the feedback observed so far. The performance of the learner is measured by regret, which is the difference between (i) the total loss of the learner; and (ii) the total loss of the best benchmark in hindsight, i.e., the total loss of the best fixed decision had one known the environment ahead of time. The goal in online decision-making is to develop algorithms whose regret grows sublinearly with the time horizon T.

Two of the most well-studied models of online decision-making are stochastic multi-armed bandits and online convex optimization. They both distill different challenges of online decision-making into a clean abstraction and have a long history of research [Robbins, 1951, Lai and Robbins, 1985, Littlestone and Warmuth, 1989, Zinkevich, 2003]. We refer the reader to excellent surveys and textbooks on these topics [Bubeck and Cesa-Bianchi, 2012, Slivkins, 2019, Lattimore and Szepesvári, 2020, Hazan, 2022, Orabona, 2019]. Here, we provide a very high-level overview.

Stochastic multi-armed bandits are the quintessential model of the exploration-exploitation trade-off. There is a finite set actions, which are commonly called arms. Each arm is associated with a fixed but unknown reward distribution. (In the bandits literature it is customary to use rewards instead of losses.) In each round, the learner chooses an arm and observes a reward drawn from the chosen arm's distribution. The goal is to minimize the difference between (i) the expected total reward of always choosing the arm with the highest mean; and (ii) the expected total reward of the algorithm. There are two main challenges. First, the learner does not know the true distributions. Second, the learner only observes *bandit feedback*, i.e., it only observes the reward of the chosen arm in a round; it does not know what would have happened had it chosen a different arm. Therefore, the learner faces a dilemma: should it choose an arm that has performed well so far (exploitation) or should it choose a different arm to learn more about its distribution (exploration)? There is a long line of work establishing tight bounds on regret and considering many extensions to this basic model. We refer the reader to the citations above for a comprehensive introduction.

Online convex optimization (OCO) is another classic model that focuses on a different challenge of online decision-making: how should a learner choose decisions when the rewards or losses can be arbitrary? The setup of OCO is as follows. (In the OCO literature it is customary to use losses instead of rewards.) There is a closed, convex set of actions. In each round, the learner chooses an action and, simultaneously, an adversary chooses a convex loss function. Then, the learner observes the loss function and suffers the loss associated with their chosen action. The goal is to minimize the difference between (i) the total loss of the algorithm; and (ii) the total loss of the best fixed action had one known the loss functions ahead of time. Note that in the OCO model, at the end of a round the learner knows the loss function, i.e., observes *full feedback*; it knows what would have happened had it chosen a different action. This is unlike stochastic bandits. However, the challenge is that the loss functions can be chosen arbitrarily. How does one use past feedback to choose decisions that have low regret with respect to the best fixed action that knows the sequence of loss functions in advance? As with stochastic bandits, there is a long line of work establishing tight bounds on regret and considering many extensions to this basic model. We refer the reader to the citations above for a comprehensive introduction.

While these classic models provide a clean abstraction for important challenges of online decision-making, a major limitation is that they lack "memory": the current reward or loss depends only on the current decision. This is not true in many applications. For example, consider dynamic pricing where a seller has an inventory of items and wants to choose a sequence of prices to maximize total revenue. A typical approach is to model this in the stochastic multi-armed bandit model, where arms correspond to prices and the reward corresponds to the revenue from a sale (if any). In this formulation the revenue in a round depends only on the current price. However, the seller may be constrained by a limited inventory of items that can run out well before the end of the time horizon. Therefore, the revenue in a round depends on the current inventory, which in turn depends on the history of prices chosen by the seller. For another example, consider online control of a physical dynamical system, such as variable-speed wind turbines in wind energy electric power production [Boukhezzar and Siguerdidjane, 2010]. The system state is the electrical and mechanical properties of the turbine system; the control inputs are generator torque and blade pitch angle, and these influence the turbine speed and the future system state; and the objective is to design a controller to maximize power generation and minimize system load. A popular approach is to model this as an OCO problem. However, note that the current system state depends on the entire history of controls. Therefore, the current loss depends not only on the current decision but the entire history of decisions. The OCO framework cannot capture such long-term dependence of the current loss on the past decisions and neither can existing generalizations that allow the loss to depend on a *constant* number of past decisions [Anava et al., 2015]. Although a series of approximation arguments can be used to apply finite memory generalizations of OCO to the online linear control problem [Agarwal et al., 2019b], there is no OCO framework that captures the complete long-term dependence of current losses on past decisions.

In the rest of this section we provide a high-level overview of our contributions in Part I of this dissertation, where we augment the aforementioned models of online decision-making with a notion of memory and present simple algorithms with strong theoretical guarantees.

Non-monotonic Resource Utilization in the Bandits with Knapsacks Problem. Bandits with knapsacks (BwK) [Badanidiyuru et al., 2018] is an influential model of online decision-making under uncertainty that incorporates resource consumption constraints. In each round, the decision-maker observes an outcome consisting of a reward and a vector of nonnegative resource consumptions, and the budget of each resource is decremented by its consumption. While BwK augments the classic stochastic multi-armed bandit model with a notion of memory through resource budgets, it has a key limitation: the budget of each resource is always non-increasing. This monotonicity assumption does not capture applications such as dynamic pricing where resources can be replenished or consumed in a non-monotonic manner.

In Chapter 3 we introduce a natural generalization of the stochastic BwK problem that allows non-monotonic resource utilization. In each round, the decision-maker observes an outcome consisting of a reward and a vector of resource *drifts* that can be positive, negative or zero, and the budget of each resource is incremented by its drift. Our main result is a Markov decision process (MDP) policy that has *constant* regret with respect to a linear programming (LP) relaxation when the decision-maker *knows* the true outcome distributions. Instead of merely sampling from the optimal probability distribution over arms, our policy samples from a perturbed distribution to ensure that the budget of each resource stays close to a decreasing sequence of thresholds. The sequence is chosen such that the expected leftover budget is a constant and proving this is a key step in the regret analysis. Our work combines aspects of related work on logarithmic regret for BwK [Flajolet and Jaillet, 2015, Li et al., 2021a]. We build upon this to develop a learning algorithm that has *logarithmic* regret with respect to the same LP relaxation when the decision-maker *does not know* the true outcome distributions. We also present a reduction from BwK to our model that shows our regret bound matches existing results [Li et al., 2021a].

Online Convex Optimization with Unbounded Memory. Online convex optimization (OCO) is a different influential model of online decision-making. In each round, the learner chooses a decision in a convex set and an adversary chooses a convex loss function, and then the learner suffers the loss associated with their current decision. However, in many applications the learner's loss depends not only on the current decision but on the entire history of decisions until that point. The OCO framework and its existing generalizations do not capture this, and they can only be applied to many settings of interest after a long series of approximation arguments. They also leave open the question of whether the dependence on memory is tight because there are no non-trivial lower bounds.

In Chapter 4 we introduce a generalization of the OCO framework, "Online Convex Optimization with Unbounded Memory", that captures long-term dependence on past decisions. We introduce the notion of p-effective memory capacity, H_p, that quantifies the maximum influence of past decisions on present losses. We prove an $O(\sqrt{H_p T})$ upper bound on the policy regret and a matching (worst-case) lower bound. As a special case, we prove the first non-trivial lower bound for OCO with finite memory [Anava et al., 2015], which could be of independent interest, and also improve existing upper bounds. We demonstrate the broad applicability of our framework by using it to derive regret bounds, and to improve and simplify existing regret bound derivations, for a variety of online learning problems including online linear control and an online variant of performative prediction.

1.2 Analyzing Interactions Offline via Network Science

In addition to learning with interactions online, a learner may also want to analyze past interactions offline. In many applications these interactions can be projected onto a weighted graph. For example, consider a user "listening session" on the music streaming platform Spotify [Kumar et al., 2020]. One simple graph that can be constructed from this interaction data is a bipartite graph: the nodes are users and songs, and there is a weighted edge between a user and a song with the weight equal to the number of times the user has listened to the song. A different graph is the following: the nodes are songs, and there is a weighted edge between two songs with the weight equal to the number of times the two songs have co-appeared in a listening session. After building such graphs from interaction data, the learner can get insights into their rich structure by using tools from network science.

In the rest of this section we provide a high-level overview of our contributions in Part II of this dissertation, where we generalize classic network tools for unweighted graphs to the weighted case.

Retrieving Top Weighted Triangles in Graphs. Small subgraph patterns, also called graphlets or network motifs, have proven fundamental for the understanding of the structure of complex networks [Milo et al., 2002, 2004, Benson et al., 2016]. One of the simplest non-trivial subgraph patterns is the triangle (3-clique) and the basic problem of triangle counting and enumeration has been studied extensively from theoretical and practical perspectives [Avron, 2010, Eden et al., 2017, Kolda et al., 2013, Berry et al., 2015, Stefani et al., 2017a]. These developments are often driven by the desire to scale graph counting to large networks, where performing computations naively is intractable. The focus on triangles is in part spurred by the widespread use of the pattern in graph mining applications, including community detection [Berry et al., 2011, Gleich and Seshadhri, 2012, Rohe and Qin, 2013], network comparison [Contractor et al., 2006, Mahadevan et al., 2007, Pržulj, 2007], representation learning [Henderson et al., 2012, Rossi and Ahmed, 2015], and generative modeling [Robins et al., 2007, Robles-Granda et al., 2016]. Additionally, triangle-based network statistics such as the clustering coefficient are used extensively in the social sciences [Durak et al., 2012, Lawrence, 2006, Burt, 2007, Welles et al., 2010].

The edge weights present in many real-world networks offer additional insight into the structure of these networks [Wasserman and Faust, 1994, Jia et al., 2019, Xu et al., 2014]. In such networks, one goal is to discover the *top-weighted triangles*, where the weight of a triangle is derived from the weights of its constituent edges. An application for finding top-weighted triangles appears in prediction tasks involving higher-order network interactions. The goal of the "higher-order link prediction" problem is to predict which new groups of nodes will simultaneously interact (such as which group of songs will co-appear in a listening session in the future) [Benson et al., 2018]. However, nearly all of the algorithmic literature on scalable counting or enumeration of triangles focuses on *unweighted* graphs.

In Chapter 5 we develop algorithms that enable the fast discovery of triangles with the largest weight in a *weighted* graph. Formally, let $G = (V, E, w)$ be a simple, undirected graph with positive edge weights w. Let the weight of a triangle in G be the generalized p-mean of its consitutent edge weights. That is, if a triangle defined by vertices a, b, and c has edge weights w_{ab}, w_{bc}, and w_{ac}, then the weight of the triangle is

$$m_p(a, b, c) = \left(\frac{1}{3}(w_{ab}^p + w_{bc}^p + w_{ac}^p)\right)^{\frac{1}{p}}.$$

Given a graph G and a positive integer k, we develop a suite of determinstic and randomized sampling algorithms to extract the top-k triangles in G. (We use top-k to refer to the triangles having the k largest weights.)

Our deterministic algorithms are optimized for extracting the top-k triangles for small k (typically up to a few tens of thousands). These algorithms take advantage of the inherent heavy-tailed edge weight distribution common in real-world networks. Our randomized sampling algorithms aim to extract a large number of heavy triangles (not necessarily the top-k). We show that this family of sampling algorithms is closely connected to the prior sampling algorithms for *counting* triangles on *unweighted* graphs [Seshadhri et al., 2014]. Furthermore, we show that these sampling algorithms are easily parallelizable.

We find that a carefully tuned parallel implementation of our deterministic algorithm performs well across a broad range of large weighted graphs, even outperforming the fast random sampling algorithms that are not guaranteed to enumerate all of the top-weighted triangles. A parallel implementation of our algorithm running on a commodity server with 64 cores can find the top 1000 weighted triangles in under 10 seconds on several graphs with hundreds of millions of weighted edges and in 30 seconds on a graph with nearly two billion weighted edges. We compare this with the off-the-shelf alternative approach, which would be an intelligent triangle enumeration algorithm that maintains a heap of the top-weighted triangles. Our proposed algorithms are orders of magnitude faster than this standard approach.

1.3 Bibliographic Notes

Chapter 3 is based on joint work with Robert Kleinberg [Kumar and Kleinberg, 2022a], Chapter 4 is based on joint work with Sarah Dean and Robert Kleinberg [Kumar et al., 2023], and Chapter 5 is based on joint work with Paul Liu, Moses Charikar and Austin Benson [Kumar et al., 2020].

Part I

Learning with Interactions via Online Decision-making

CHAPTER 2

PRELIMINARIES FOR ONLINE DECISION-MAKING

2.1 A Framework for Online Decision-making

In this section we introduce a unifying framework for online decision-making that includes the problems from Chapters 3 and 4 as special cases. First, we present many formal and abstract definitions. Then, we instantiate the framework for specific problems to illustrate how to use it concretely. We note that other frameworks exist in the literature, e.g., Kleinberg [2005], Lattimore and Szepesvári [2020], Foster et al. [2022], and our presentation is influenced by them. Our goal is to establish notation and terminology that we will use in the rest of this dissertation.

Throughout Part I, we use $T \in \mathbb{N}$ to denote the *time horizon*, i.e., the length of the interaction between a learner and the environment.

Definition 2.1.1 (Domain)**.** The domain of an online decision-making problem is a tuple of sets $(\mathcal{X}, \Psi, \mathcal{L}, \mathcal{G})$, where $\mathcal{X}$ denotes the underlying action space, Ψ denotes the state space, $\mathcal{L}$ denotes the space of loss functions from Ψ to $\Delta(\mathbb{R})$, and $\mathcal{G}$ denotes the space of dynamics functions from $\Psi \times \mathcal{X}$ to $\Delta(\Psi)$.[1]

Definition 2.1.2 (Feedback)**.** The feedback space of an online decision-making problem is a tuple of sets $\Phi = (\Phi_l, \Phi_g)$ representing the possible values of the loss feedback and dynamics feedback in a single round. The feedback model of

[1]We use $\Delta(\cdot)$ to denote the set of probability distributions over a set.

an online decision-making problem is a tuple of functions $\phi = (\phi_l, \phi_g)$ representing the feedback models for loss and dynamics respectively, where $\phi_l : \mathcal{L} \times \Psi \to \Phi_l$ and $\phi_g : \mathcal{G} \times \Psi \times \mathcal{X} \to \Phi_g$.

Two common examples of feedback are *full feedback* and *bandit feedback*. For example, consider the feedback for loss functions. (Similar definitions hold for the dynamics functions.) If, in every round, the learner observes the full function l, i.e., $\Phi_l = \mathcal{L}$ and $\phi_l(l, \psi) = l$, then this is called full feedback. If, in every round, the learner only observes the value of the loss function evaluated at the current state, i.e., $\Phi_l = \mathbb{R}$ and $\phi_l(l, \psi) = l(\psi)$, then this is called bandit feedback.

Definition 2.1.3 (Decision/Policy)**.** A decision or policy $\pi \in \Pi$ for an online decision-making problem is a function $\pi : \cup_{t=1}^{\infty} \Phi_g^{t-1} \to \Delta(\mathcal{X})$ that maps a history of dynamics feedback to a distribution over actions.

Note that the input to a policy is a variable-length sequence of dynamics feedback. We will restrict ourselves to "causal" policies, i.e., in round t the input to a policy is the dynamics feedback from the first $t-1$ rounds. In particular, in round t the input to a policy only depends on *past* feedback.

Definition 2.1.4 (Algorithm)**.** An algorithm for an online decision-making problem is a sequence of functions $\mathsf{ALG} = (\mathsf{ALG}_t)_{t=1}^{T}$, where $\mathsf{ALG}_t : (\Phi_l \times \Phi_g)^{t-1} \to \Delta(\Pi)$.

Informally, an algorithm for an online decision-making problem produces a sequence of policies (or decisions), $(\pi_1, \ldots, \pi_T)$, where π_t depends only on the loss and dynamics feedback from the first $t-1$ rounds. The policy π_t produces an action, denoted by x_t, that is a function of the dynamics feedback from the first $t-1$ rounds. One might wonder: why does the algorithm not choose an action x_t directly? This can be recovered as a special case by setting $\Pi = \mathcal{X}$. That is, by considering history-independent, constant-action policies where each $\pi \in \Pi$ corresponds to a fixed action $x \in \mathcal{X}$. However, for some problems, such as bandits with knapsacks (Chapter 3), online linear control (Chapter 4), etc., it is crucial to consider general, history-dependent policies.

Definition 2.1.5 (Environment)**.** An environment for an online decision-making problem is a sequence of functions $\mathsf{ENV} = (\mathsf{ENV}_t)_{t=1}^{T}$, where $\mathsf{ENV}_t : \mathcal{X}^{t-1} \to \Delta(\mathcal{L}\times\mathcal{G})$. Furthermore,

- An environment is an oblivious adversary if ENV_t is a constant function. That is, (l_t, g_t) is a sample from an arbitrary distribution over $\mathcal{L} \times \mathcal{G}$, but it does not depend on the learner's actions.
- An environment is stochastic if $\mathsf{ENV}_t = \mathsf{ENV}_1$ for all $t \in [T]$ and ENV_1 is a point mass on $\mathcal{L} \times \mathcal{G}$. That is, there is $(l, g) \in \mathcal{L} \times \mathcal{G}$ such that $(l_t, g_t) = (l, g)$ for all $t \in [T]$.

Informally, an environment for an online decision-making problem produces a sequence of loss and dynamics functions, $((l_1, g_1), \ldots, (l_T, g_T))$, where (l_t, g_t) depends only on the learner's actions in the first $t-1$ rounds. An environment is an oblivious adversary if the sequence does not depend on the learner's actions.

Note that this is still a non-trivial problem because the adversary can choose an arbitrary point mass on the space of loss and dynamics functions and this point mass can differ across rounds. An environment is stochastic if the loss and dynamics functions are the same across all rounds.

Definition 2.1.6 (Online Decision-making Problem)**.** An online decision-making problem is a tuple $\mathsf{ODM} = (\mathcal{X}, \Psi, \mathcal{L}, \mathcal{G}, \Phi, \phi, \Pi, \mathsf{ENV})$.

In summary, an online decision-making problem proceeds as follows. The initial state $\psi_0 \in \Psi$ is known to the learner. In each round $t \in [T]$,

1. The algorithm chooses a policy $\pi_t \in \Pi$, and the environment simultaneously chooses a loss function $l_t \in \mathcal{L}$ and a dynamics function $g_t \in \mathcal{G}$.
   - The algorithm chooses $\pi_t \sim \mathsf{ALG}_t((\phi_l(l_i, \psi_i), \phi_g(g_i, \psi_{i-1}, x_i))_{i=1}^{t-1})$.
   - The policy chooses $x_t \sim \pi_t((\phi_g(g_i, \psi_{i-1}, x_i))_{i=1}^{t-1})$.
   - The environment chooses $(l_t, g_t) \sim \mathsf{ENV}_t((x_i)_{i=1}^{t-1})$.
2. The state updates to $\psi_t \sim g_t(\psi_{t-1}, x_t)$.
3. The learner suffers loss $l_t^{\mathsf{ALG}} \sim l_t(\psi_t)$.
4. The learner observes feedback $(\phi_l(l_t, \psi_t), \phi_g(g_t, \psi_{t-1}, x_t)) \in \Phi_l \times \Phi_g$.

Note that in our framework the loss depends on the state, rather than the action or the state-action pair. However, this can be encoded by expanding the definition of state to include the action. This should become clear from the examples in the subsection that follows.

The performance of an algorithm for an online decision-making problem is measured by regret. It is the difference between two quantities: (i) the expected total loss of the algorithm; and (ii) the expected total loss of a fixed policy.

Definition 2.1.7 (Regret)**.** Let ALG be an algorithm for an online decision-making problem $\mathsf{ODM} = (\mathcal{X}, \Psi, \mathcal{L}, \mathcal{G}, \Phi, \phi, \Pi, \mathsf{ENV})$. Then, the regret is defined as

$$\mathsf{Reg}_T^{\mathsf{ODM}}(\mathsf{ALG}) = \mathbb{E}\left[\sum_{t=1}^{T} l_t^{\mathsf{ALG}}\right] - \inf_{\pi^* \in \Pi} \mathbb{E}\left[\sum_{t=1}^{T} l_t^{\pi^*}\right],$$

where (i) $l_t^{\pi^*} \sim l_t(\psi_t^{\pi^*})$ and $\psi_t^{\pi^*}$ denotes the state in round t as a result of using the algorithm $(\pi^*, \ldots, \pi^*)$; and (ii) the expectation is with respect to the randomness in $\mathsf{ALG}, \mathsf{ENV}, \pi_t, l_t$, and g_t.

2.1.1 Examples

Stochastic multi-armed bandits. The stochastic multi-armed bandits problem [Robbins, 1951, Lai and Robbins, 1985, Bubeck and Cesa-Bianchi, 2012, Slivkins, 2019, Lattimore and Szepesvári, 2020] is defined as follows. There is a finite set of k arms denoted by $\mathcal{X}_{\text{mab}}$. In each round $t \in [T]$, the learner chooses an arm from $\mathcal{X}_{\text{mab}}$ and observes a reward $r_t \in [0, 1]$. Each arm has a reward distribution and the reward in round t, r_t, is drawn from the chosen arm's reward distribution. The goal of the learner is to minimize the regret, i.e., the difference between the total expected reward of the best fixed arm and the total expected reward of the algorithm. This can be formulated in the online decision-making framework defined above as follows.

- Actions $\mathcal{X} = \mathcal{X}_{\text{mab}}$.
- States $\Psi = \mathcal{X}_{\text{mab}}$.
- Dynamics function $g_t = g$ for all rounds, where $g(\psi_{t-1}, x_t) = x_t$. Here, $\mathcal{G}$ consists of a single function defined by the previous equation.
- Loss function $l_t = l$ for all rounds, where $l(x) = -r(x)$ and $r(x)$ is a reward sampled from the reward distribution of arm x. Here, $\mathcal{L}$ is the set of all mappings from $\mathcal{X}_{\text{mab}}$ to $\Delta([-1, 0])$.
- Feedback spaces $\Phi_l = [-1, 0]$ and $\Phi_g = \mathcal{X}_{\text{mab}}$.
- Feedback functions $\phi_l(l, \psi_t) = l(\psi_t)$ and $\phi_g(g, \psi_{t-1}, x_t) = x_t$.
- Decisions Π are constant functions taking values in the set of point-mass distributions over $\mathcal{X}_{\text{mab}}$.
- The environment ENV is stochastic.

Online convex optimization. The online convex optimization problem [Littlestone and Warmuth, 1989, Zinkevich, 2003, Hazan, 2022, Orabona, 2019] is defined as follows. There is a closed, convex set of actions denoted by $\mathcal{X}_{\text{oco}}$. In each round $t \in [T]$, the learner chooses an action from $\mathcal{X}_{\text{oco}}$ and, simultaneously, an oblivious adversary chooses a convex loss function $f_t : \mathcal{X}_{\text{oco}} \to \mathbb{R}$. The learner observes the loss function and suffers the loss associated with their chosen action. The goal of the learner is to minimize the regret, i.e., the difference between the total loss of the algorithm and the total loss of the best fixed action. This can be formulated in the online decision-making framework defined above as follows.

- Actions $\mathcal{X} = \mathcal{X}_{\text{oco}}$.
- States $\Psi = \mathcal{X}_{\text{oco}}$.
- Dynamics function $g_t = g$ for all rounds, where $g(\psi_{t-1}, x_t) = x_t$. Here, $\mathcal{G}$ consists of a single function defined by the previous equation.
- Loss function $l_t = f_t$ for all rounds. Here, $\mathcal{L}$ is the set of all convex functions from $\mathcal{X}_{\text{oco}}$ to $\mathbb{R}$.
- Feedback spaces $\Phi_l = \mathcal{L}$ and $\Phi_g = \mathcal{X}_{\text{oco}}$.
- Feedback functions $\phi_l(l_t, \psi_t) = l_t$ and $\phi_g(g, \psi_{t-1}, x_t) = x_t$.
- Decisions Π are constant functions taking values in the set of point-mass distributions over $\mathcal{X}_{\text{oco}}$.
- The environment ENV is an oblivious adversary.

Online linear control with adversarial disturbances. Online linear control (OLC) is the problem of controlling a system with linear dynamics, adversarial disturbances, and adversarial and convex losses. It combines aspects from control theory and online learning. We refer the reader to Agarwal et al. [2019b] for more details. Here, we introduce the basic mathematical setup of the problem.

Let $\mathcal{S} \subseteq \mathbb{R}^{d_s}$ and $\mathcal{U} \subseteq \mathbb{R}^{d_u}$ denote the state and control spaces. Let s_t and u_t denote the state and control at time t with s_1 being the initial state. Let Π_{olc} denote a given class of controllers.[2] In each round $t \in [T]$, the learner chooses a controller π_t (which yields a control u_t) and, simultaneously, an oblivious adversary chooses (i) a disturbance $w_t \in \mathbb{R}^{d_s}$ with $\|w_t\|_2 \leq W$ and (ii) a convex loss function $c_t : \mathcal{S} \times \mathcal{U} \to [0, 1]$. The system evolves according to linear dynamics $s_{t+1} = F s_t + G u_t + w_t$, where $F \in \mathbb{R}^{d_s \times d_s}$ and $G \in \mathbb{R}^{d_u \times d_u}$ are matrices known to the learner. The learner observes the loss function c_t, the new state s_{t+1}, and suffers the loss $c_t(s_t, u_t)$. The goal of the learner is to minimize the regret with respect to the given class of controllers Π_{olc}. This can be formulated in the online decision-making framework defined above as follows.

- Actions $\mathcal{X} = \mathcal{U}$.
- States $\Psi = \mathcal{S} \times \mathcal{U} \times \mathcal{S}$.
- Intial state $\psi_0 = (s_1, 0)$.
- Dynamics function g_t defined as

$$g_t((s_{t-1}, u_{t-1}, s_t), u_t) = (s_t, u_t, F s_t + G u_t + w_t).$$

  Here, $\mathcal{G}$ is the set of all mappings from $(\mathcal{S} \times \mathcal{U} \times \mathcal{S}) \times \mathcal{U}$ to $\mathcal{S} \times \mathcal{U} \times \mathcal{S}$ that are defined by the previous equation, and parameterized by F, G and w, where $F, G \in \mathbb{R}^{d_s \times d_s}$ are *fixed* and $w \in \mathbb{R}^{d_s}$.

[2]A controller chooses the control as a function of the past states, e.g., $u_t = K s_t$ for a "linear controller" $K \in \mathbb{R}^{d_u \times d_s}$.

- Loss function $l_t((s_t, u_t, s_{t+1})) = c_t(s_t, u_t)$. Here, $\mathcal{L}$ is the set of all convex functions from $\mathcal{S} \times \mathcal{U} \times \mathcal{S}$ to $[0, 1]$.
- Feedback spaces $\Phi_l = \mathcal{L}$ and $\Phi_g = \mathcal{S} \times \mathcal{U} \times \mathcal{S}$.
- Feedback functions $\phi_l(l_t, (s_t, u_t, s_{t+1})) = l_t$ and $\phi_g(g_t, (s_{t-1}, u_{t-1}, s_t), u_t) = g_t((s_{t-1}, u_{t-1}, s_t), u_t)$.
- Decisions $\Pi = \Pi_{\text{olc}}$. For concreteness and $\mathcal{X} = \mathbb{R}^{d_u}$, if Π_{olc} is the class of linear controllers, then each policy $\pi \in \Pi_{\text{olc}}$ is parameterized by a matrix $K_\pi \in \mathbb{R}^{d_u \times d_s}$ and produces $\pi((s_j, u_j, s_{j+1})_{j=1}^{t}) = K_\pi s_t$.
- The environment ENV is an oblivious adversary.

Instead of solving this problem directly, which leads to nonconvexity, we solve it by formulating it as an online convex optimization with unbounded memory problem, which we develop in Chapter 4.

2.2 Standard Analysis of Follow-the-Regularized-Leader

In this section we state and prove some existing results about the follow-the-regularized-leader (FTRL) algorithm [Shalev-Shwartz and Singer, 2006, Abernethy et al., 2008]. These results are well known in the literature. But we use them heavily in Chapter 4, so we prove them here for completeness. While we formulated online convex optimization (OCO) in our online decision-making framework in the previous section, here we simplify the notation and setup (e.g., ignore the dynamics because they are trivial for this problem). This is so that we can focus on the details of the FTRL algorithm and its regret analysis without unnecessary clutter.

Consider the following setup for an OCO problem. Let T denote the time horizon. Let the action space $\mathcal{X}_{\text{oco}}$ be a closed, convex subset of a Hilbert space and $f_t : \mathcal{X}_{\text{oco}} \to \mathbb{R}$ be loss functions chosen by an oblivious adversary. The functions f_t are convex and L_f-Lipschitz continuous. The game between the learner and the adversary proceeds as follows. In each round $t \in [T]$, the learner chooses $x_t \in \mathcal{X}_{\text{oco}}$ and the learner suffers loss $f_t(x_t)$. The goal of the learner is to minimize regret,

$$\mathsf{Reg}_T^{\mathsf{OCO}}(\mathsf{FTRL}) = \sum_{t=1}^{T} f_t(x_t) - \min_{x \in \mathcal{X}_{\text{oco}}} \sum_{t=1}^{T} f_t(x). \tag{2.1}$$

Let $R : \mathcal{X}_{\text{oco}} \to \mathbb{R}$ be an α-strongly convex regularizer satisfying $|R(x) - R(\tilde{x})| \leq D$ for all $x, \tilde{x} \in \mathcal{X}_{\text{oco}}$. The FTRL algorithm chooses iterates x_t as

$$x_t \in \arg\min_{x \in \mathcal{X}_{\text{oco}}} \sum_{s=1}^{t-1} f_s(x) + \frac{R(x)}{\eta}, \tag{2.2}$$

where η is a tunable parameter referred to as the step-size. The following theorem provides an upper bound on $\mathsf{Reg}_T^{\mathsf{OCO}}(\mathsf{FTRL})$.

Theorem 2.1. *FTRL (Eq. (2.2)) satisfies*

$$\|x_{t+1} - x_t\|_{\mathcal{X}_{oco}} \leq \eta \frac{L_f}{\alpha} \quad \text{and} \quad \mathsf{Reg}_T^{\mathsf{OCO}}(\mathsf{FTRL}) \leq \frac{D}{\eta} + \eta \frac{T L_f^2}{\alpha},$$

where $\|\cdot\|_{\mathcal{X}_{oco}}$ denotes the norm associated with $\mathcal{X}_{oco}$. Choosing $\eta = \sqrt{\frac{\alpha D}{T L_f^2}}$ yields

$$\mathsf{Reg}_T^{\mathsf{OCO}}(\mathsf{FTRL}) \leq O\left(\sqrt{\frac{D}{\alpha} T L_f^2}\right).$$

We first state and prove two lemmas and then end this subsection by using them to prove the above theorem. In what follows, let $f_0 = \frac{R}{\eta}$. The analysis in this section closely follows Karlin [2017].

Lemma 2.2.1. *For all $x \in \mathcal{X}_{oco}$, FTRL (Eq. (2.2)) satisfies*

$$\sum_{t=0}^{T} f_t(x) \geq \sum_{t=0}^{T} f_t(x_{t+1}).$$

Proof. We use proof by induction on T. The base case is $T = 0$. By definition, $x_1 \in \arg\min_{x \in \mathcal{X}_{\text{oco}}} R(x)$. Therefore, $R(x) \geq R(x_1)$ for all $x \in \mathcal{X}_{\text{oco}}$. Recalling the notation $f_0 = \frac{R}{\eta}$ proves the base case. Now, assume that the lemma is true for $T-1$. That is,

$$\sum_{t=0}^{T-1} f_t(x) \geq \sum_{t=0}^{T-1} f_t(x_{t+1}).$$

Let $x \in \mathcal{X}_{\text{oco}}$ be arbitrary. Since $x_{T+1} \in \arg\min_{x \in \mathcal{X}_{\text{oco}}} \sum_{t=0}^{T} f_t(x)$, we have

$$\begin{aligned}
\sum_{t=0}^{T} f_t(x) &\geq \sum_{t=0}^{T} f_t(x_{T+1}) \\
&= \sum_{t=0}^{T-1} f_t(x_{T+1}) + f_T(x_{T+1}) \\
&\geq \sum_{t=0}^{T-1} f_t(x_{t+1}) + f_T(x_{T+1}) && \text{by inductive hypothesis} \\
&= \sum_{t=0}^{T} f_t(x_{t+1}).
\end{aligned}$$

This completes the proof. ■

Lemma 2.2.2. *For all $x \in \mathcal{X}_{oco}$, FTRL (Eq. (2.2)) satisfies*

$$\sum_{t=1}^{T} f_t(x_t) - \sum_{t=1}^{T} f_t(x) \leq \frac{D}{\eta} + \sum_{t=1}^{T} f_t(x_t) - f_t(x_{t+1}).$$

Proof. Note that

$$\sum_{t=1}^{T} f_t(x_t) - \sum_{t=1}^{T} f_t(x) = \underbrace{f_0(x) - f_0(x_1)}_{(a)} + \underbrace{\sum_{t=1}^{T} f_t(x_t) - \sum_{t=1}^{T} f_t(x_{t+1})}_{(b)} + \underbrace{\sum_{t=0}^{T} f_t(x_{t+1}) - \sum_{t=0}^{T} f_t(x)}_{(c)}.$$

Term (a) is bounded above by $\frac{D}{\eta}$ from using $f_0 = \frac{R}{\eta}$ and the definition of D. Term (c) is at most 0 from using Lemma 2.2.1. This yields the desired inequality. ■

Now, we restate Theorem 2.1 and prove it using the above two lemmas.

Theorem 2.1. *FTRL (Eq. (2.2)) satisfies*

$$\|x_{t+1} - x_t\|_{\mathcal{X}_{oco}} \leq \eta \frac{L_f}{\alpha} \quad \textit{and} \quad \mathsf{Reg}_T^{\mathsf{OCO}}(\mathsf{FTRL}) \leq \frac{D}{\eta} + \eta \frac{TL_f^2}{\alpha},$$

where $\|\cdot\|_{\mathcal{X}_{oco}}$ *denotes the norm associated with* $\mathcal{X}_{oco}$*. Choosing* $\eta = \sqrt{\frac{\alpha D}{TL_f^2}}$ *yields*

$$\mathsf{Reg}_T^{\mathsf{OCO}}(\mathsf{FTRL}) \leq O\left(\sqrt{\frac{D}{\alpha} TL_f^2}\right).$$

Proof. Let $x^* \in \arg\min_{x \in \mathcal{X}_{\text{oco}}} \sum_{t=1}^{T} f_t(x)$. Using Lemma 2.2.2 we have

$$\sum_{t=1}^{T} f_t(x_t) - \sum_{t=1}^{T} f_t(x^*) \leq \frac{D}{\eta} + \sum_{t=1}^{T} f_t(x_t) - f_t(x_{t+1}). \tag{2.3}$$

We can bound the summands in the sum above as follows. Define $f_{0:t-1}(x) = \sum_{s=0}^{t-1} f_s(x)$. Then, $x_t \in \arg\min_{x \in \mathcal{X}_{\text{oco}}} f_{0:t-1}(x)$. and $x_{t+1} \in \arg\min_{x \in \mathcal{X}_{\text{oco}}} f_{0:t}(x)$. Since $\{f_s\}_{s=1}^{T}$ are convex, R is α-strongly-convex, and $f_0 = \frac{R}{\eta}$, we have that $f_{0:t-1}$ is $\frac{\alpha}{\eta}$-strongly-convex. So,

$$f_{0:t-1}(x_{t+1}) \geq f_{0:t-1}(x_t) + \frac{\alpha}{2\eta}\|x_{t+1} - x_t\|_{\mathcal{X}_{\text{oco}}}^2,$$
$$f_{0:t}(x_t) \geq f_{0:t}(x_{t+1}) + \frac{\alpha}{2\eta}\|x_{t+1} - x_t\|_{\mathcal{X}_{\text{oco}}}^2.$$

Adding the above two inequalities yields

$$f_t(x_t) - f_t(x_{t+1}) \geq \frac{\alpha}{\eta} \|x_{t+1} - x_t\|^2_{\mathcal{X}_{\text{oco}}}. \tag{2.4}$$

Since f_t are convex and L_f-Lipschitz continuous, we also have

$$f_t(x_t) - f_t(x_{t+1}) \leq L_f \|x_{t+1} - x_t\|_{\mathcal{X}_{\text{oco}}}. \tag{2.5}$$

Combining Eqs. (2.4) and (2.5) we have

$$\|x_{t+1} - x_t\|_{\mathcal{X}_{\text{oco}}} \leq \eta \frac{L_f}{\alpha}.$$

This proves the first part of the theorem. Now, using this in Eq. (2.5) we have

$$f_t(x_t) - f_t(x_{t+1}) \leq \eta \frac{L_f^2}{\alpha}. \tag{2.6}$$

Finally, substituting this in Eq. (2.3) proves the second part of the theorem. ■

2.3 Concentration Inequalities

We end this chapter with concentration inequalities that we use in this dissertation. While there exist more general statements of these results, we tailor the presentation for our purposes. The first inequality is Hoeffding's inequality that bounds the tail probability of sub-Gaussian random variables, i.e., random variables whose tail probability is bounded by a Gaussian. For our purposes, it suffices to consider the special case of bounded random variables.

Lemma 2.3.1 (Hoeffding's Inequality)**.** *Let $(Z_i)_{i=1}^m$ be independent and identically distributed (i.i.d.) random variables with $Z_i \in [a, b]$ and $\mathbb{E}[Z_i] = \mu$. Then,*

$$\mathbf{Pr}\left[\sum_{i=1}^m Z_i < w\right] \le \exp\left(-\frac{2(w - m\mu)^2}{m(b-a)^2}\right).$$

In particular, if $m \ge \frac{2w}{\mu}$, then $w = \alpha\, m\mu$ for $0 < \alpha \le \frac{1}{2}$ and

$$\mathbf{Pr}\left[\sum_{i=1}^m Z_i < w\right] \le \exp\left(-\frac{2\mu^2 m(\alpha - 1)^2}{(b-a)^2}\right) \le \exp\left(-\frac{\mu^2 m}{2(b-a)^2}\right).$$

This inequality is standard throughout the literature and a proof can be found in many textbooks, e.g., Wainwright [2019, Proposition 2.5]. Sometimes we encounter a sequence of random variables that is neither independent nor identically distributed. However, if it is a submartingale or supermartingale difference sequence, then its tail probability can be bounded using the Azuma-Hoeffding inequality. For our purposes, it suffices to consider the submartingale case. Informally, a sequence of random variables is a submartingale difference sequence if the expected value of a random variable conditioned on the past is non-negative. We make this precise in the following definitions before stating the Azuma-Hoeffding inequality.

Definition 2.3.1 (Filtration)**.** Let $(\Omega, \mathcal{F}, P)$ be a probability space. We say that the sequence $(\mathcal{F}_i)_{i=1}^\infty$ is a filtration if, for all $i \ge 1$, (i) $\mathcal{F}_i$ is a sub-σ-algebra of $\mathcal{F}$; and (ii) $\mathcal{F}_i \subseteq \mathcal{F}_{i+1}$.

Definition 2.3.2 (Submartingale Difference Sequence)**.** Let $(\Omega, \mathcal{F}, P)$ be a probability space and $(\mathcal{F}_i)_{i=1}^\infty$ be a filtration. Let $(Z_i)_{i=1}^\infty$ be a sequence of random variables adapted to $(\mathcal{F}_i)_{i=1}^{\infty}$, i.e., Z_i is $\mathcal{F}_i$-measurable for all $i \geq 1$. We say that $((Z_i, \mathcal{F}_i))_{i=1}^{\infty}$ is a submartingale difference sequence if $\mathbb{E}[Z_i \mid \mathcal{F}_{i-1}] \geq 0$ for all $i \geq 1$.

Lemma 2.3.2 (Azuma-Hoeffding Inequality)**.** *Let $((Z_i, \mathcal{F}_i))_{i=1}^{\infty}$ be a submartingale difference sequence that satisfies $Z_i \in [-n, n]$ and $\mathbb{E}[Z_i] \geq \mu > 0$. Then, $((Z_i - \mu, \mathcal{F}_i))_{i=1}^{\infty}$ is also a submartingale difference sequence and*

$$\mathbf{Pr}\left[\sum_{i=1}^{m} Z_i \leq 0\right] = \mathbf{Pr}\left[\sum_{i=1}^{m} Z_i - \mu < -m\mu\right] \leq \exp\left(-\frac{\mu^2 m}{4n^2}\right).$$

This inequality is also standard and a proof can be found in many textbooks, e.g., Wainwright [2019, Corollary 2.20]. The next lemma provides a bound on the tail probability of the sum of random variables whose tail probabilities are subexponential. We are not aware of a statement in this form in the literature, so we also provide a proof for completeness.

Lemma 2.3.3. *Let $(Z_i)_{i=1}^{m}$ be i.i.d. random variables with $Z_i \in \mathbb{Z}_+$, $\mathbb{E}[Z_i] = \mu > 0$, and*

$$\mathbf{Pr}[Z_i \geq z] \leq c' \exp\left(-\frac{z}{c}\right)$$

for some constants $c, c' > 0$. If $t \geq 4mcc'$, then

$$\mathbf{Pr}\left[\sum_{i=1}^{m} Z_i \geq t\right] \leq \exp\left(-\frac{t}{4c}\right).$$

Proof. Let Z denote a random variable with the same distribution as Z_1. We can upper bound its moment generating function as

$$\begin{aligned}
\mathbb{E}\left[\exp\left(sZ\right)\right] &= \int_0^\infty \mathbf{Pr}\left[\exp\left(sZ\right) \geq t\right] dt \\
&= \int_0^\infty \mathbf{Pr}\left[Z \geq \frac{\ln t}{s}\right] dt \\
&\leq c' \int_0^\infty \exp\left(-\frac{\ln t}{sc}\right) dt \\
&= c' \int_0^\infty t^{-\frac{1}{sc}} dt \\
&= c' \left(1 - \frac{1}{sc}\right)^{-1} \left[t^{\left(1-\frac{1}{sc}\right)}\right]_0^\infty \\
&= c' \frac{sc}{1 - sc} && \text{if } sc < 1 \\
&\leq 2scc' && \text{if } sc \leq \frac{1}{2} \\
&\leq \exp\left(2scc'\right). && (2.7)
\end{aligned}$$

Let $S_m = \sum_{i=1}^m Z_i$. Using Chernoff's bound, we have

$$\begin{aligned}
\mathbf{Pr}\left[S_m \geq t\right] &= \mathbf{Pr}\left[\exp(sS_m) \geq \exp(st)\right] \\
&\leq \frac{\mathbb{E}\left[\exp(sS_m)\right]}{\exp(st)} \\
&= \frac{\mathbb{E}\left[\exp(s\sum_{i=1}^m Z_i)\right]}{\exp(st)} \\
&= \frac{\prod_{i=1}^m \mathbb{E}\left[\exp(sZ_i)\right]}{\exp(st)} && \text{since } Z_i \text{ are i.i.d.} \\
&\leq \exp\left(2mscc' - st\right) && \text{from Eq. (2.7)} \\
&\leq \exp\left(\frac{2st}{4} - st\right) && \text{since } t \geq 4mcc' \\
&= \exp\left(-\frac{st}{2}\right) \\
&= \exp\left(-\frac{t}{4c}\right),
\end{aligned}$$

where the last equality follows by choosing $s = \frac{1}{2c}$. ■

We end this section with a statement of Khintchine's inequality that suffices for our purposes. A stronger statement and a proof can be found in, e.g., Wolff [2003, Proposition 4.5].

Lemma 2.3.4 (Khintchine's Inequality). *Let $(Z_i)_{i=1}^m$ be i.i.d. random variables with* $\mathbf{Pr}[Z_i = 1] = \mathbf{Pr}[Z_i = -1] = \frac{1}{2}$. *Let $x_1, \ldots, x_m \in \mathbb{R}$. Then,*

$$\left(\sum_{i=1}^m |x_i|^2\right)^{\frac{1}{2}} \leq \left(\mathbb{E}\left|\sum_{i=1}^m Z_i x_i\right|\right).$$

CHAPTER 3

NON-MONOTONIC RESOURCE UTILIZATION IN THE BANDITS WITH KNAPSACKS PROBLEM

In this chapter we study a generalization of the classic stochastic multi-armed bandit problem, namely, bandits with knapsacks. Bandits with knapsacks (BwK) [Badanidiyuru et al., 2018] is an influential model of online decision-making under uncertainty that incorporates resource consumption constraints. In each round, the decision-maker observes an outcome consisting of a reward and a vector of nonnegative resource consumptions, and the budget of each resource is decremented by its consumption. In this chapter we introduce a natural generalization of the stochastic BwK problem that allows non-monotonic resource utilization. In each round, the decision-maker observes an outcome consisting of a reward and a vector of resource *drifts* that can be positive, negative or zero, and the budget of each resource is incremented by its drift. Our main result is a Markov decision process (MDP) policy that has *constant* regret against a linear programming (LP) relaxation when the decision-maker *knows* the true outcome distributions. We build upon this to develop a learning algorithm that has *logarithmic* regret against the same LP relaxation when the decision-maker *does not know* the true outcome distributions. We also present a reduction from BwK to our model that shows our regret bound matches existing results [Li et al., 2021a]. This chapter is based on joint work with Robert Kleinberg [Kumar and Kleinberg, 2022a].

3.1 Introduction

Multi-armed bandits are the quintessential model of online decision-making under uncertainty in which the decision-maker must trade-off between exploration and exploitation. They have been studied extensively and have numerous applications, such as clinical trials, ad placements, and dynamic pricing to name a few. We refer the reader to Bubeck and Cesa-Bianchi [2012], Slivkins [2019], Lattimore and Szepesvári [2020] for an introduction to bandits. An important shortcoming of the basic stochastic bandits model is that it does not take into account resource consumption constraints that are present in many of the motivating applications. For example, in a dynamic pricing application the seller may be constrained by a limited inventory of items that can run out well before the end of the time horizon. The bandits with knapsacks (BwK) model [Tran-Thanh et al., 2010, 2012, Badanidiyuru et al., 2013, 2018] remedies this by endowing the decision-maker with some initial budget for each of m resources. In each round, the outcome is a reward and a vector of nonnegative resource consumptions, and the budget of each resource is decremented by its consumption. The process ends when the budget of any resource becomes nonpositive. However, even this formulation fails to model that in many applications resources can get replenished or renewed over time. For example, in a dynamic pricing application a seller may receive shipments that increase their inventory level.

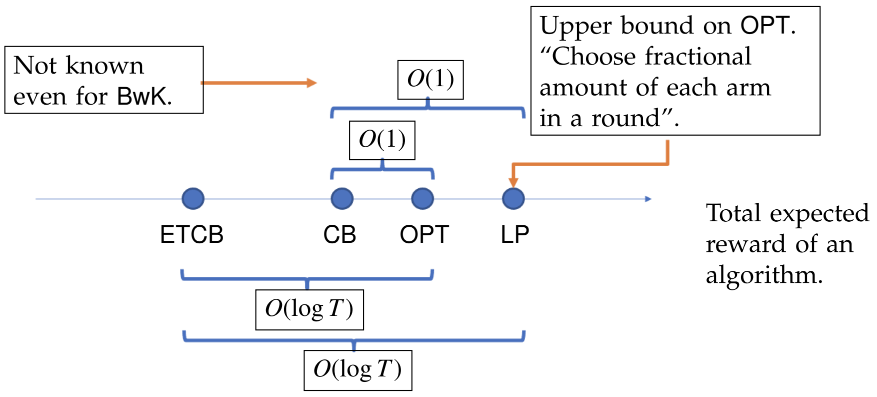

Not known even for BwK.
$O(1)$
Upper bound on OPT. "Choose fractional amount of each arm in a round".
$O(1)$
ETCB
CB
OPT
LP
Total expected reward of an algorithm.
$O(\log T)$
$O(\log T)$

Figure 3.1: Our MDP policy (CB) has constant regret with respect to a linear programming relaxation (LP) and hence, with respect to the optimal MDP policy. Our learning algorithm (ETCB) has logarithmic regret with respect to the same relaxation.

Contributions In this chapter we introduce a natural generalization of BwK by allowing non-monotonic resource utilization. The decision-maker starts with some initial budget for each of m resources. In each round, the outcome is a reward and a vector of resource *drifts* that can be positive, negative or zero, and the budget of each resource is incremented by its drift. A negative drift has the effect of decreasing the budget akin to consumption in BwK and a positive drift has the effect of increasing the budget. We consider two settings: (i) when the decision-maker *knows* the true outcome distributions and must design a Markov decision process (MDP) policy; and (ii) when the decision-maker *does not know* the true outcome distributions and must design a learning algorithm.

Our main contribution is an MDP policy, ControlBudget (CB), that has *constant* regret with respect to a linear programming (LP) relaxation. Such a result was not known even for BwK. We build upon this to develop a learning algorithm, ExploreThenControlBudget (ETCB), that has *logarithmic* regret with respect to the same LP relaxation. We also present a reduction from BwK to our model and show that our regret bound matches existing results.

Instead of merely sampling from the optimal probability distribution over arms, our policy samples from a perturbed distribution to ensure that the budget of each resource stays close to a decreasing sequence of thresholds. The sequence is chosen such that the expected leftover budget is a constant and proving this is a key step in the regret analysis. Our work combines aspects of related work on logarithmic regret for BwK [Flajolet and Jaillet, 2015, Li et al., 2021a].

Related Work Multi-armed bandits have a rich history and logarithmic instance-dependent regret bounds have been known for a long time [Lai and Robbins, 1985, Auer et al., 2002a]. Since then, there have been numerous papers extending the stochastic bandits model in a variety of ways [Auer et al., 2002b, Slivkins, 2014, Kleinberg et al., 2019, Badanidiyuru et al., 2018, Immorlica et al., 2022, Agrawal and Devanur, 2014, 2016].

To the best of our knowledge, there are three papers on logarithmic regret bounds for BwK. Flajolet and Jaillet [2015] showed the first logarithmic regret bound for BwK. In each round, their algorithm finds the optimal basis for an optimistic version of the LP relaxation, and chooses arms from the resulting basis to ensure that the average resource consumption stays close to a pre-specified level. Even though their regret bound is logarithmic in T and inverse linear in the suboptimality gap, it is exponential in the number of resources. Li et al. [2021a] showed an improved logarithmic regret bound that is polynomial in the number of resources, but it scales inverse quadratically with the suboptimality gap and their definition of the gap is different from the one in Flajolet and Jaillet [2015]. The main idea behind improving the dependence on the number of resources is to proceed in two phases: (i) identify the set of arms and binding resources in the optimal solution; (ii) in each round, solve an adaptive, optimistic version of the LP relaxation and sample an arm from the resulting probability distribution. Finally, Sankararaman and Slivkins [2021] show a logarithmic regret bound for BwK with respect to a *fixed-distribution* benchmark. However, the regret of this benchmark itself with the optimal MDP policy can be as large as $O(\sqrt{T})$ [Flajolet and Jaillet, 2015, Li et al., 2021a].

3.2 Preliminaries

3.2.1 Model

Let T denote a finite time horizon, $\mathcal{X} = \{1, \ldots, k\}$ a set of k arms, $\mathcal{J} = \{1, \ldots, m\}$ denote a set of m resources, and $B_{0,j} = B$ denote the initial budget of resource j. In each round $t \in [T]$, if the budget of any resource is less than 1, then $\mathcal{X}_t = \{1\}$. Otherwise, $\mathcal{X}_t = \mathcal{X}$. The algorithm chooses an arm $x_t \in \mathcal{X}_t$ and observes an outcome $o_t = (r_t, d_{t,1}, \ldots, d_{t,m}) \in [0, 1]{\times}[-1, 1]^m$. The algorithm earns reward r_t and the budget of resource $j \in \mathcal{J}$ is incremented by drift $d_{t,j}$ as $B_{t,j} = B_{t-1,j} + d_{t,j}$.

Each arm $x \in \mathcal{X}$ has an outcome distribution over $[0,1] \times [-1,1]^m$ and o_t is drawn from the outcome distribution of the arm x_t. We use $\mu_x^o = (\mu_x^r, \mu_x^{d,1}, \ldots, \mu_x^{d,m})$ to denote the expected outcome vector of arm x consisting of the expected reward and the expected drifts for each of the m resources.[1] We also use $\mu^{d,j} = (\mu_x^{d,j} : x \in \mathcal{X})$ to denote the vector of expected drifts for resource j. We assume that arm $x^0 = 1 \in \mathcal{X}$ is a null arm with three important properties: (i) its reward is zero a.s.; (ii) the drift for each resource is nonnegative a.s.; and (iii) the expected drift for each resource is positive. The second and third properties of the null arm plus the model's requirement that $x_t = 1$ if $\exists j$ s.t. $B_{t-1,j} < 1$ ensure that the budgets are nonnegative a.s. and can be safely increased from 0.

Our model is intended to capture applications featuring resource renewal, such as the following. In each round, each resource gets replenished by some random amount and the chosen arm consumes some random amount of each resource. If the consumption is less than replenishment, the resource gets renewed. The random variable $d_{t,j}$ then models the net replenishment minus consumption. The full model presented above is more general because it allows both the consumption and replenishment to depend on the arm pulled.

We consider two settings in this chapter.

[1]In fact, our proofs remain valid even if the outcome distribution depends on the past history provided the conditional expectation is independent of the past history and fixed for each arm. In this case μ_x^o denotes the conditional expectation of o_t when arm x is pulled in round t. Since our proofs rely on the Azuma-Hoeffding inequality (Lemma 2.3.2), we need this assumption on the conditional expectation to hold.

MDP setting The decision-maker *knows* the true outcome distributions. In this setting the model implicitly defines an MDP, where the state is the budget vector, the actions are arms, and the transition probabilities are defined by the outcome distributions of the arms.

Learning setting The decision-maker *does not know* the true outcome distributions.

The goal is to design to an MDP policy for the first setting and a learning algorithm for the second, and bound their regret against an LP relaxation as defined in the next subsection.

Formulation as an online decision-making problem. Before continuing, we take a brief detour to show that the problem defined above can be formulated in the online decision-making framework (Definition 2.1.6), which is defined by the tuple $(\mathcal{X}, \Psi, \mathcal{L}, \mathcal{G}, \Phi, \phi, \Pi, \mathsf{ENV})$.

- Actions $\mathcal{X}$ are the same as above.
- States $\Psi = \mathbb{R}_+^m \times \mathcal{X}$.
- Initial state $\psi_0 = ((B)_{j=1}^m, x^0)$.
- Dynamics function $g_t = g$ for all rounds, where

$$g(\psi_{t-1}, x_t) = g(((B_{t-1,j})_{j=1}^m, x_{t-1}), x_t) = ((B_{t-1,j} + d_{t,j})_{j=1}^m, x_t) = ((B_{t,j})_{j=1}^m, x_t).$$

Here, $(d_{t,j})_{j=1}^{m}$ are drifts sampled from the drift distribution of arm x_t. Here, $\mathcal{G}$ is the set of all mappings from $(\mathbb{R}_{+}^{m} \times \mathcal{X}) \times \mathcal{X}$ to $\Delta(\mathbb{R}_{+}^{m}) \times \mathcal{X}$ that are defined by the previous equation, and parameterized by all mappings from $\mathcal{X}$ to $\Delta([-1, 1]^m)$ representing the drift distributions of the arms.

- Loss function $l_t = l$ for all rounds, where $l(\psi_t) = l((B_{t,j})_{j=1}^{m}, x_t) = -r_t$. Here, r_t is the reward sampled from the reward distribution of arm x_t. Here, $\mathcal{L}$ is the set of all mappings from $\mathcal{X}$ to $\Delta([-1, 0])$.
- Feedback spaces $\Phi_l = [-1, 0]$ and $\Phi_g = \Psi$.
- Feedback functions $\phi_l(l, \psi_t) = l(\psi_t)$ and $\phi_g(g, \psi_{t-1}, x_t) = g(\psi_{t-1}, x_t)$.
- Decisions Π is the set of all policies with the constraint that they pull the null arm when the budget of any resource less than 1. That is, mappings π from the history of dynamics feedback, $h = (((B_{s,j})_{j=1}^{m}, x_s))_{s=1}^{t}$, to $\Delta(\mathcal{X})$ such that $\pi(h)$ is a point-mass distribution on x^0 if $\exists j$ such that $B_{t,j} < 1$.
- The environment ENV is stochastic.

For simplicity, in the rest of this chapter we will use the formalism and notation defined previously instead of the one above.

3.2.2 Linear Programming Relaxation

Similar to Badanidiyuru et al. [2018, Lemma 3.1], we consider the following LP relaxation that provides an upper bound on the total expected reward of any algorithm:

$$\mathsf{OPT}_{\mathsf{LP}} = \max_{p} \left\{ \sum_{x\in\mathcal{X}} p_x \mu_x^r : \sum_{x\in\mathcal{X}} p_x \mu_x^{d,j} \geq -B/T \ \forall j \in \mathcal{J}, \ \sum_{x\in\mathcal{X}} p_x = 1, \ p_x \geq 0 \ \forall x \in \mathcal{X} \right\}. \quad (3.1)$$

Lemma 3.2.1. *The total expected reward of any algorithm is at most* $T \cdot \mathsf{OPT}_{\mathsf{LP}}$.

The proof of this lemma, similar to those in existing works [Agrawal and Devanur, 2014, Badanidiyuru et al., 2018], follows from the observations that (i) the variables $p = \{p_x : x \in \mathcal{X}\}$ can be interpreted as the probability of choosing arm x in a round; and (ii) if we set p_x equal to the expected number of times x is chosen by an algorithm divided by T, then it is a feasible solution for the LP.

Definition 3.2.1 (Regret)**.** The regret of an algorithm $\mathcal{A}$ is defined as

$$\mathsf{Reg}_T^{\mathsf{BwK}}(\mathcal{A}) = T \cdot \mathsf{OPT}_{\mathsf{LP}} - \mathsf{REW}(\mathcal{A}),$$

where $\mathsf{REW}(\mathcal{A})$ denotes the total expected reward of $\mathcal{A}$.

3.2.3 Assumptions

We assume that the initial budget of every resource is $B \leq T$. This assumption is without loss of generality because otherwise we can scale the drifts by dividing them by the smallest budget. This results in a smaller support set for the drift distribution that is still contained in $[-1, 1]$.

Our assumptions about the null arm x^0 are a major difference between our model and BwK. In BwK the budgets can only decrease and the process ends when the budget of any resource reaches 0. However, in our model the budgets can increase or decrease, and the process ends at the end of the time horizon. Our assumptions about the null arm allow us to increase the budget from 0 without making it negative.[2] A side-effect of this is that in our model we can even assume that B is a small constant because we can always increase the budget by pulling the null arm, in contrast to existing literature on BwK that assume the initial budgets are large and often scale with the time horizon.

A standard assumption for achieving logarithmic regret in stochastic bandits is that the gap between the total expected reward of an optimal arm and that of a second-best arm is positive. There are a few different ways in which one could translate this to our model where the optimal solution is a mixture over arms. We make the following choice. We assume that there exists a unique set of arms X^* that form the support set of the LP solution and a unique set of resources J^* that correspond to binding constraints in the LP solution [Li et al., 2021a]. We define the gap of the problem instance in Definition 3.4.1 and our uniqueness assumption[3] implies that the gap is strictly positive.

We make a few separation assumptions parameterized by four positive constants that can be arbitrarily small. First, the smallest magnitude of the drifts, $\delta_{\text{drift}} = \min\{|\mu_x^{d,j}| : x \in \mathcal{X}, j \in \mathcal{J}\}$, satisfies $\delta_{\text{drift}} > 0$. Second, the smallest sin-

[2]In a model where, in each round, each resource gets replenished by some random amount and the chosen arm consumes some random amount of each resource, the null arm represents the option to remain idle and do nothing while waiting for resource replenishment. See Section 3.10 for more discussion on the assumptions about the null arm.

[3]This assumption is essentially without loss of generality because the set of problem instances with multiple optimal solutions is a set of measure zero.

gular value of the LP constraint matrix, denoted by $\sigma_{\min}$, satisfies $0 < \sigma_{\min} < 1$. Third, the LP solution p^* satisfies $p^*_x \geq \delta_{\text{support}} > 0$ for all $x \in X^*$. Fourth, $\sum_{x \in X^*} p^*_x \mu^{d,j}_x \geq \delta_{\text{slack}} > 0$ for all resources $j \notin J^*$. The first assumption is necessary for logarithmic regret bounds because otherwise one can show that the regret of the optimal algorithm for the case of one resoure and one zero-drift arm is $\Theta(\sqrt{T})$ (Section 3.11). The second and third assumptions are essentially the same as in existing literature on logarithmic regret bounds for BwK [Flajolet and Jaillet, 2015, Li et al., 2021a]. The fourth assumption allows us to design algorithms that can increase the budgets of the non-binding resources away from 0, thereby reducing the number of times the algorithm has to pull the null arm. Otherwise, if they have zero drift, then, as stated above, the regret of the optimal algorithm for the case of one resource and one zero-drift arm is $\Theta(\sqrt{T})$ (Section 3.11).

3.3 MDP Policy with Constant Regret

In this section we design an MDP policy, ControlBudget (Algorithm 2), with constant regret in terms of T for the setting when the learner knows the true outcome distributions and our model implicitly defines an MDP (Section 3.2.1). At a high level, ControlBudget, which shares similarities with Flajolet and Jaillet [2015, Algorithm UCB-Simplex], plays arms to keep the budgets close to a decreasing sequence of thresholds. The choice of this sequence allows us to show that the expected leftover budgets and the expected number of null arm pulls are constants.

This is a key step in proving the final regret bound. We start by considering the special case of one resource in Section 3.3.1 because it provides intuition for the general case of multiple resources in Section 3.3.2. In this section, we present algorithms and results with proof sketches. We defer detailed proofs to later subsections.

3.3.1 Special Case: One Resource

Since there is only one resource we drop the superscript j in this section. We say that an arm x is a positive (resp. negative) drift arm if $\mu_x^d > 0$ (resp. $\mu_x^d < 0$). The following lemma characterizes the possible solutions of the LP (Eq. (3.1)).

Lemma 3.3.1. *The solution of the LP relaxation (Eq. (3.1)) is supported on at most two arms. Furthermore, if $T \geq {}^{B}/_{\delta_{drift}}$, then the solution belongs to one of three categories: (i) supported on a single positive drift arm; (ii) supported on the null arm and a negative drift arm; (iii) supported on a positive drift arm and a negative drift arm.*

The proof of this lemma follows from properties of LPs and a case analysis of which constraints are tight. Our MDP policy, ControlBudget (Algorithm 1), deals with the three cases separately and satisfies the following regret bound.[4]

[4]In this theorem and the rest of the chapter, we use $\tilde{C}$ to denote a constant that depends on problem parameters, including k, m, and the various separation constants mentioned in Section 3.2.3, but *does not depend on* T. We use this notation because the main focus of this work is how the regret scales as a function of T.

Algorithm 1: ControlBudget (for $m = 1$)

```
Input: time horizon $T$, initial budget $B$, set of arms $\mathcal{X}$, set of resources $\mathcal{J}$,
       constant $c > 0$.
1  Set $B_0 = B$.
2  if LP solution is supported on positive drift arm $x^p$ then
3      for $t = 1, 2, \ldots, T$ do
4          If $B_{t-1} < 1$, pull $x^0$. Otherwise, pull $x^p$.
5      end
6  else if LP solution is supported on null arm $x^0$ and negative drift arm $x^n$ then
7      for $t = 1, 2, \ldots, T$ do
8          Define threshold $\tau_t = c \log(T - t)$.
9          If $B_{t-1} < \max\{1, \tau_t\}$, pull $x^0$. Otherwise, pull $x^n$.
10     end
11 else if LP solution is supported on positive drift arm $x^p$ and negative drift arm $x^n$
    then
12     for $t = 1, 2, \ldots, T$ do
13         Define threshold $\tau_t = c \log(T - t)$.
14         If $B_{t-1} < 1$, pull $x^0$. If $1 \le B_{t-1} < \tau_t$, pull $x^p$. Otherwise, pull $x^n$.
15     end
```

Theorem 3.1. *If $c \geq \frac{12}{\delta^2_{drift}}$, the MDP policy* ControlBudget *(Algorithm 1) satisfies*

$$\mathsf{Reg}_T^{\mathsf{BwK}}(\mathsf{ControlBudget}) \leq \tilde{C},$$

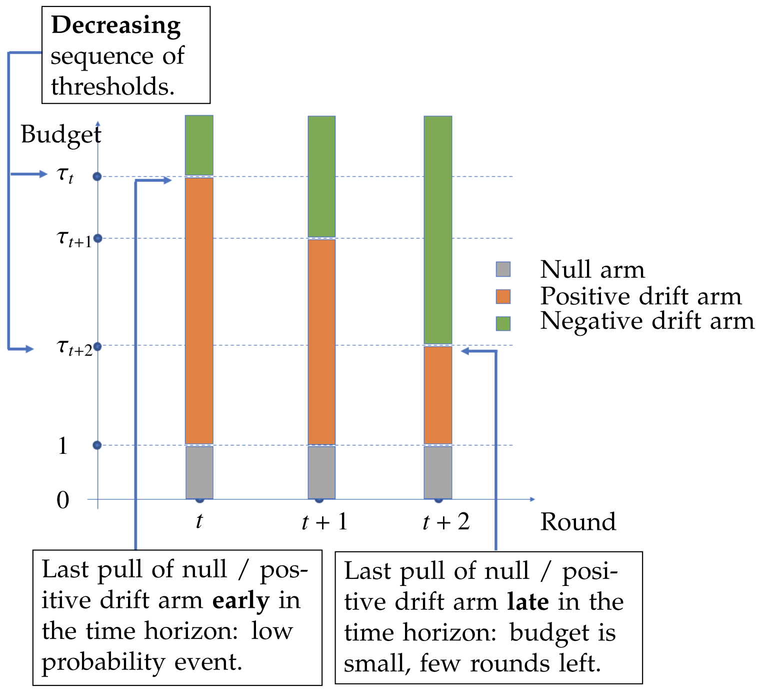
where $\tilde{C} = O\left(\delta_{drift}^{-4} \ln\left(\left(1 - \exp\left(-\frac{\delta^2_{drift}}{8}\right)\right)^{-1}\right) + \delta_{drift}^{-1}\left(1 - \exp\left(\delta^2_{drift}\right)\right)^{-2}\right)$ is a constant.

We defer all proofs in this subsection to Section 3.7. The proof of Theorem 3.1 follows from the following sequence of lemmas.

Lemma 3.3.2. *If the LP solution is supported on a positive drift arm x^p, then*

$$\mathsf{Reg}_T^{\mathsf{BwK}}(\mathsf{ControlBudget}) \leq \tilde{C},$$

where $\tilde{C} = O\left(\delta_{drift}^{-3} \ln\left(\left(1 - \exp\left(-\frac{\delta^2_{drift}}{8}\right)\right)^{-1}\right)\right)$ is a constant.



Decreasing sequence of thresholds.
Budget
τ_t
τ_{t+1}
τ_{t+2}
1
0
t
$t+1$
$t+2$
Round
Null arm
Positive drift arm
Negative drift arm
Last pull of null / positive drift arm **early** in the time horizon: low probability event.
Last pull of null / positive drift arm **late** in the time horizon: budget is small, few rounds left.

Figure 3.2: An illustration of ControlBudget for one resource when the optimal LP solution is supported on a positive drift arm and a negative drift arm. The algorithm maintains a decreasing sequence of thresholds $\{\tau_t\}$. If the budget is above τ_t, it plays the negative drift arm (indicated by green in the figure). If the budget is below the τ_t but above 1, it plays the positive drift arm (indicated by orange in the figure). Otherwise, it plays the null arm (indicated by grey in the figure). Since the thresholds are **decreasing**, the algorithm is more aggressive about using the negative drift arm as it approaches the end of the time horizon.

We can write the regret in terms of the norm of $\xi = (\xi_{x^p})$, where ξ_{x^p} is the expected difference between the number of times x^p is played by the LP and by ControlBudget. This is equal to the expected number of times the policy plays the null arm and, in turn, is equal to the expected number of rounds in which the budget is below 1. Since both x^0 and x^p have positive drift, this is a transient random walk that drifts away from 0. It is known that such a walk spends a constant number of rounds in any state in expectation.

Lemma 3.3.3. *If the LP solution is supported on the null arm x^0 and a negative drift arm x^n, then*

$$\mathsf{Reg}_T^{\mathsf{BwK}}(\mathsf{ControlBudget}) \leq \tilde{C} \cdot \mathbb{E}[B_T],$$

where $\tilde{C} = O(\delta_{drift}^{-1})$ is a constant.

We can write the regret in terms of the norm of $\xi = (\xi_{x^0}, \xi_{x^n})$, where ξ_x is the expected difference between the number of times x is played by the LP and by ControlBudget. Since both constraints (resource and sum-to-one) are tight, the lemma follows by writing $\xi = D^{-1}b$ and taking norms, where D is the LP constraint matrix and $b = (-\mathbb{E}[B_T], 0)$.

Lemma 3.3.4. *If the LP solution is supported on a positive drift arm x^p and a negative drift arm x^n, then*

$$\mathsf{Reg}_T^{\mathsf{BwK}}(\mathsf{ControlBudget}) \leq \tilde{C} \cdot \max\{\mathbb{E}[B_T], \mathbb{E}[N_{x^0}]\},$$

where $\mathbb{E}[N_{x^0}]$ denotes the expected number null arm pulls and $\tilde{C} = O(\delta_{drift}^{-1})$ is a constant.

This lemma follows similarly to the previous one by writing regret in terms of the norm of $\xi = (\xi_{x^p}, \xi_{x^n})$ and writing $\xi = D^{-1}b$ for $b = (-\mathbb{E}[B_T], \mathbb{E}[N_{x^0}])$.

Therefore, proving that $\mathsf{Reg}_T^{\mathsf{BwK}}(\mathsf{ControlBudget})$ is a constant in T requires proving that both the expected leftover budget and expected number of null arm pulls are constants. Intuitively, we could ensure $\mathbb{E}[B_T]$ is small by playing the negative drift arm whenever the budget is at least 1. However, there is constant probability of the budget decreasing below 1 and the expected number of null arm pulls becomes $O(T)$. ControlBudget solves the tension between the two objectives by carefully choosing a decreasing sequence of thresholds τ_t. The threshold is initially far from 0 to ensure low probability of pre-mature resource depletion, but decreases to 0 over time to ensure small expected leftover budget and decreases at a rate that ensures the expected number of null arm pulls is a constant.

Lemma 3.3.5. *If the LP solution is supported on a positive drift arm x^p and a negative drift arm x^n, and $c \geq \frac{12}{\delta_{drift}^2}$, then*

$$\mathbb{E}[N_{x^0}] \leq \tilde{C},$$

where $\tilde{C} = O\left(\delta_{drift}^{-3} \ln\left(\left((1 - \exp\left(-\frac{\delta_{drift}^2}{8}\right)\right)^{-1}\right)\right)$ is a constant.

If the budget is below the threshold, i.e., $B_{t-1} < \tau_t$ for some t, then ControlBudget pulls x^p until $B_s \geq \tau_{s+1}$ for some $s \geq t$. Since x^p has positive drift, the event that repeated pulls decrease the budget towards 0 is a low probability event. Using this, our choice of $\tau_t = c \log(T - t)$ for an appropriate constant c, and summing over all rounds shows that the expected number of rounds in which the budget is less than 1 is a constant in T.

Lemma 3.3.6. *If the LP solution is supported on two arms, and $c \geq \frac{12}{\delta_{drift}^2}$, then*

$$\mathbb{E}[B_T] \leq \tilde{C},$$

where $\tilde{C} = \tilde{O}\left(\left(1 - \exp\left(\delta_{drift}^2\right)\right)^{-2} + \delta_{drift}^{-2}\right)$ is a constant.

If $B_{t-1} \geq \tau_t$, then ControlBudget pulls a negative drift arm x^n. We can upper bound the expected leftover budget by conditioning on q, the number of consecutive pulls of x^n at the end of the timeline. The main idea in completing the proof is that (i) if q is large, then it corresponds to a low probability event; and (ii) if q is small, then the budget in round $T - q$ was smaller than τ_q, which is a decreasing sequence in q, and there are few rounds left so the budget cannot increase by too much.

3.3.2 General Case: Multiple Resources

Now we use the ideas from Section 3.3.1 to tackle the case of $m > 1$ resources that is much more challenging. Generalizing Lemma 3.3.1, the solution of the LP relaxation (Eq. (3.1)) is supported on at most $\min\{k, m\}$ arms. Informally, our MDP policy, ControlBudget (Algorithm 2), samples an arm from a probability distribution that ensures drifts bounded away from 0 in the "correct directions": (i) a binding resource j has drift at least γ_t if $B_{t-1,j} < \tau_t$ and drift at most $-\gamma_t$ if $B_{t-1,j} \geq \tau_t$; and (ii) a non-binding resource j has drift at least $\frac{1}{2}\gamma_t$ if $B_{t-1,j} < \tau_t$. This allows us to show that the expected leftover budget for each binding resource and the expected number of null arm pulls are constants in terms of T.

Theorem 3.2. *If $c \geq \frac{12}{\gamma^{*2}}$, the regret of* ControlBudget *(Algorithm 2) satisfies*

$$\mathsf{Reg}_T^{\mathsf{BwK}}(\mathsf{ControlBudget}) \leq \tilde{C},$$

Algorithm 2: ControlBudget (for general m)

```
   Input: time horizon $T$, initial budget $B$, set of arms $\mathcal{X}$, set of resources $\mathcal{J}$,
          constant $c > 0$.
1  Set $B_{0,j} = B$ for all $j \in \mathcal{J}$.
2  Define threshold $\tau_t = c \log(T - t)$.
3  for $t = 1, 2, \dots, T$ do
4      if $\exists j \in \mathcal{J}$ such that $B_{t-1,j} < 1$ then
5          Pull the null arm $x^0$.
6      else
7          Define $D$ to be the square submatrix of the LP constraint matrix,
             whose columns correspond to the arms $X^*$ and whose rows
             correspond to the binding constraints in the LP solution. Let the
             first $|X^*| - 1$ rows correspond to binding budget constraints and the
             last row correspond to the sum-to-1 constraint. Define the vector $b$
             to be the right-hand side of the LP corresponding the rows in $D$.
8          Define $s_t \in \{\pm 1\}^{|X^*|-1} \times 0$ as follows. Let $j$ denote the resource
             corresponding to row $i \in [|X^*| - 1]$ in the matrix $D$ and vector $b$.
             Then, the $i$th entry of $s_t$ is $+1$ if $B_{t-1,j} < \tau_t$ and $-1$ otherwise.
9          Define $\gamma_t$ to be the solution to the following constrained
             optimization problem:

             $$\max_{\gamma \in [0,1]} \left\{ \gamma : p = D^{-1}(b + \gamma s_t) \geq 0,\ p^T \mu^{d,j} \geq \frac{\gamma}{2}\ \forall j \in \mathcal{J} \setminus J^* \text{ if } B_{t-1,j} < \tau_t \right\}. \quad (3.2)$$

10         Sample an arm from the probability distribution $p_t = D^{-1}(b + \gamma_t s_t)$.
11     end
12 end
```

where γ^* (defined in Lemma 3.3.9) and $\tilde{C}$ are constants with

$$\tilde{C} = O\left(m\sigma_{\min}^{-1}\left(m(\gamma^*)^{-3} \ln\left(\left((1 - \exp\left(-\gamma^{*2}\right)\right)^{-1}\right) + \left(1 - \exp(\gamma^{*2})\right)^{-2}\right)\right).$$

We defer all proofs in this subsection to Section 3.8. The proof of Theorem 3.2 follows from the following sequence of lemmas. The next two lemmas are generalizations of Lemmas 3.3.3 and 3.3.4 with essentially the same proofs. Recall J^*

denotes the unique set of resources that correspond to binding constraints in the LP solution (Section 3.2.3).

Lemma 3.3.7. *If the LP solution includes the null arm x^0 in its support, then*

$$\mathsf{Reg}_T^{\mathsf{BwK}}(\mathsf{ControlBudget}) \le \tilde{C} \cdot \left(\sum_{j \in J^*} \mathbb{E}[B_{T,j}] \right),$$

where $\tilde{C} = O(\sigma_{\min}^{-1})$ is a constant.

Lemma 3.3.8. *If the LP solution does not include the null arm x^0 in its support, then*

$$\mathsf{Reg}_T^{\mathsf{BwK}}(\mathsf{ControlBudget}) \le \tilde{C} \cdot \left(\sum_{j \in J^*} \mathbb{E}[B_{T,j}] + \mathbb{E}[N_{x^0}] \right),$$

where $\mathbb{E}[N_{x^0}]$ denotes the expected number of null arm pulls and $\tilde{C} = O(\sigma_{\min}^{-1})$ is a constant.

Lemmas 3.3.10 and 3.3.11 are generalizations of Lemmas 3.3.5 and 3.3.6 with similar proofs after taking a union bound over resources. But we first need Lemma 3.3.9 that lets us conclude there is drift of magnitude at least $\gamma^* > 0$ in the "correct directions" as stated earlier.

Lemma 3.3.9. *[Flajolet and Jaillet, 2015, Lemma 14] In each round t, $\gamma_t \ge \gamma^* = \frac{\sigma_{\min} \min\{\delta_{support}, \delta_{slack}\}}{4m}$.*

The proof of this lemma is identical to Flajolet and Jaillet [2015, Lemma 14] but we provide a proof in Section 3.8 for completeness.

Lemma 3.3.10. *If the LP solution does not include the null arm in its support, then*

$$\mathbb{E}[N_{x^0}] \le \tilde{C},$$

where $\tilde{C} = O\left(m(\gamma^*)^{-3} \ln \left(\left((1 - \exp\left(-\frac{\gamma^{*2}}{8} \right) \right)^{-1} \right) \right)$ is a constant.

Lemma 3.3.11. *If the LP solution is supported on more than one arm, then for all $j \in J^*$*

$$\mathbb{E}[B_{T,j}] \leq \tilde{C},$$

where $\tilde{C} = \tilde{O}\left(\left(1 - \exp(\gamma^{*2})\right)^{-2} + (\gamma^*)^{-2}\right)$ is a constant.

A subtle but important point is that the regret analysis does not require ControlBudget to know the true expected drifts in order to find the probability vector p_t. It simply requires the algorithm to know X^*, J^*, and find any probability vector p_t that ensures drifts bounded away from 0 in the "correct directions" as stated earlier. We use this property in our learning algorithm, ExploreThenControlBudget (Algorithm 3), in the next section.

3.4 Learning Algorithm with Logarithmic Regret

In this section we design a learning algorithm, ExploreThenControlBudget (Algorithm 3), with logarithmic regret in terms of T for the setting when the learning does not know the true distributions. Our algorithm, which can be viewed as combining aspects of Li et al. [2021a, Algorithm 1] and Flajolet and Jaillet [2015, Algorithm UCB-Simplex], proceeds in three phases. It uses phase one of Li et al. [2021a, Algorithm 1] to identify the set of optimal arms X^* and the set of binding constraints J^* by playing arms in a round-robin fashion, and using confidence intervals and properties of LPs. This is reminiscent of successive elimination [Even-Dar et al., 2002], except that the algorithm tries to identify the optimal arms instead of eliminating suboptimal ones. In the second phase the algorithm continues playing the arms in X^* in a round-robin fashion to shrink the confidence radius further. In the third phase the algorithm plays a variant of the MDP policy ControlBudget (Algorithm 2) with a slighly different optimization problem for γ_t because it only has empirical esitmates of the drifts. In this section, we present algorithms and results with proof sketches. We defer detailed proofs to later subsections.

3.4.1 Additional Notation and Preliminaries

For all arms $x \in \mathcal{X}$ and rounds $t \geq k$, define the upper confidence bound (UCB) of the expected outcome vector μ_x^o as $\mathsf{UCB}_t(x) = \bar{o}_t(x) + \mathsf{rad}_t(x) \cdot \vec{1}$, where $\vec{1}$ denotes the all-ones vector, $\mathsf{rad}_t(x) = \sqrt{8 n_t(x)^{-1} \log T}$ denotes the confidence radius, $n_t(x)$ denotes the number of times x has been played before t, and $\bar{o}_t(x) = n_t(x)^{-1} \sum_t o_t \mathbb{1}[x_t = x]$ denotes the empirical mean outcome vector of x. The lower confidence bound (LCB) is defined similarly as $\mathsf{LCB}_t(x) = \bar{o}_t(x) - \mathsf{rad}_t(x) \cdot \vec{1}$.

For all arms $x \in \mathcal{X}$, let OPT_{-x} denote the value of the LP relaxation (Eq. (3.1)) with the additional constraint $p_x = 0$, and for all resources $j \in \mathcal{J}$, let OPT_{-j} denote the value when the objective has an extra $-\sum_x p_x \mu_x^{d,j} + {}^{B}/_{T}$ term [Li et al., 2021a]. Intuitively, these represent how important it is to play arm x or make the resource constraint for j a binding constraint. Define the UCB of OPT_{-x} to be the value of the LP when the expected outcome is replaced by its UCB, and denote this by

$\mathsf{UCB}_t(\mathsf{OPT}_{-x})$. The LCB for OPT_{-x}, and UCB and LCB for OPT_{-j} and $\mathsf{OPT}_{\mathsf{LP}}$ are defined similarly.

Definition 3.4.1 (Gap [Li et al., 2021a])**.** The gap of the problem instance is defined as

$$\Delta = \min\left\{\min_{x\in X^*}\{\mathsf{OPT}_{\mathsf{LP}} - \mathsf{OPT}_{-x}\}, \min_{j\notin J^*}\left\{\mathsf{OPT}_{\mathsf{LP}} - \mathsf{OPT}_{-j}\right\}\right\}.$$

3.4.2 Learning Algorithm and Regret Analysis

Theorem 3.3. *If $c \geq \frac{12}{\gamma^*}$, the regret of* ExploreThenControlBudget *(Algorithm 3) satisfies*

$$\mathsf{Reg}_T^{\mathsf{BwK}}(\mathsf{ExploreThenControlBudget}) \leq \tilde{C}\cdot\log T,$$

where γ^* (defined in Lemma 3.3.9), $\tilde{C}'$ and $\tilde{C}$ are constants with $\tilde{C}'$ denoting the constant in Theorem 3.2 and

$$\tilde{C} = O\left(\frac{km^2}{\min\{\delta_{drift}^2, \sigma_{\min}^2\}\Delta^2} + k(\gamma^*)^{-2} + \tilde{C}'\right).$$

We defer all proofs in this subsection to Section 3.9. We refer to the three while loops of ExploreThenControlBudget (Algorithm 3) as the three phases. The lemmas and corollaries following the theorem below show that the first phase consists of at most logarithmic number of rounds. It is easy to see that the second phase consists of at most logarithmic number of rounds. The third phase plays a variant of the MDP policy ControlBudget (Algorithm 2). There exists a feasible solution to the optimization problem in Line 17 that ensures drifts bounded away from 0 in the

Algorithm 3: ExploreThenControlBudget

```
Input: time horizon $T$, initial budget $B$, set of arms $\mathcal{X}$, set of resources $\mathcal{J}$,
       constant $c > 0$.
1  Set $B_{0,j} = B$ for all $j \in \mathcal{J}$.
2  Initialize $t = 1, X^* = \emptyset, J' = \emptyset$.
3  while $t < T - k$ and $|X^*| + |J'| < m + 1$ do
4      Play each arm in $\mathcal{X} \setminus \{x^0\}$ in a round-robin fashion. Play $x^0$ if $\exists j$ such
         that $B_{t-1,j} < 1$.
5      For each $x \in \mathcal{X}$, if $\mathsf{UCB}_t(\mathsf{OPT}_{-x}) < \mathsf{LCB}_t(\mathsf{OPT}_{\mathsf{LP}})$, then add $x$ to $X^*$.
6      For each $j \in \mathcal{J}$, if $\mathsf{UCB}_t(\mathsf{OPT}_{-j}) < \mathsf{LCB}_t(\mathsf{OPT}_{\mathsf{LP}})$, then add $j$ to $J'$.
7  end
8  Set $J^* = \mathcal{J} \setminus J'$.
9  while $t < T - |X^*|$ and $n_t(x) < \frac{32 \log T}{\gamma^{*2}}$ for all $x \in X^*$, where $\gamma^*$ is defined
     in Lemma 3.3.9 do
10     Play each arm in $X^*$ in a round-robin fashion.
11 end
12 while $t < T$ do
13     if $\exists j \in \mathcal{J}$ such that $B_{t-1,j} < 1$ then
14         Pull the null arm $x^0$.
15     else
16         Define $s_t$ as in Algorithm 2.
17         Choose $(\gamma_t, p_t) = \max_{\gamma \in [0,1]} \gamma$ such that there exists a probability
             vector $p$ satisfying

               $\mathsf{LCB}_t(p^T \mu^{d,j}) \geq \frac{\gamma}{8}\ \forall j \in \mathcal{J} \setminus J^*$ if $B_{t-1,j} < \tau_t$,    (3.3)
               $\mathsf{LCB}_t(p^T \mu^{d,j}) \geq \frac{\gamma}{8}\ \forall j \in J^*$ if $B_{t-1,j} < \tau_t$    (3.4)
               $\mathsf{UCB}_t(p^T \mu^{d,j}) \leq -\frac{\gamma}{8}\ \forall j \in J^*$ if $B_{t-1,j} \geq \tau_t$.    (3.5)

18         Sample an arm from the probability distribution $p_t$.
19     end
20 end
```

"correct directions" (Section 3.9). Combining the analysis of the first two phases with Theorem 3.2 lets us conclude that ExploreThenControlBudget has logarithmic regret.

Definition 3.4.2 (Clean Event)**.** The clean event is the event that for all $x \in \mathcal{X}$ and all $t \geq k$, (i) $\mu^o_x \in [\mathsf{LCB}_t(x), \mathsf{UCB}_t(x)]$; and (ii) after the first n pulls of the null arm the sum of the drifts for each resource is at least w, where $w = \frac{4096km^2 \log T}{\delta^2_{\text{drift}}\Delta^2}$ and $n = \min_{j \in \mathcal{J}} \left\{ \frac{2w}{\mu^{d,j}_{x0}} \right\}$.

Lemma 3.4.1. *The clean event occurs with probability at least* $1 - 5kmT^{-2}$.

The lemma follows from the Azuma-Hoeffding inequality (Lemma 2.3.2). Since the complement of the clean event contributes $O(kmT^{-1})$ to the regret, it suffices to bound the regret conditioned on the clean event.

Lemma 3.4.2. *If the clean event occurs, then* $\mathsf{UCB}_t(\mathsf{OPT}_{\mathsf{LP}}) - \mathsf{LCB}_t(\mathsf{OPT}_{\mathsf{LP}}) \leq \frac{8m}{\sigma_{\min}} \mathsf{rad}_t$*. A similar statement is true for* OPT_{-x} *and* OPT_{-j} *for all* $x \in \mathcal{X}$ *and* $j \in \mathcal{J}$*.*

The proof follows from a perturbation analysis of the LP and uses the confidence radius to bound the perturbations in the rewards and drifts.

Corollary 3.4.1. *If the clean event occurs and* $n_t(x) > \frac{2048m^2 \log T}{\sigma^2_{\min}\Delta^2}$ *for all* $x \in \mathcal{X}$*, then*

$$\mathsf{UCB}_t(\mathsf{OPT}_{-x}) < \mathsf{LCB}_t(\mathsf{OPT}_{\mathsf{LP}}) \textit{ and } \mathsf{UCB}_t(\mathsf{OPT}_{-j}) < \mathsf{LCB}_t(\mathsf{OPT}_{\mathsf{LP}})$$

for all $x \in X^*$ *and* $j \in \mathcal{J} \setminus J^*$*.*

This follows from substituting the bound on $n_t(x)$ into the definition of $\mathsf{rad}_t(x)$ and applying Lemma 3.4.2. In the worst case, each pull of an arm can cause the budget to drop below 1, but the clean event implies that the first n pulls of x^0 have enough total drift to allow $n_t(x)$ pulls of each non-null arm in phase 1. This allows us to upper bound the duration of phase 1 as follows.

Corollary 3.4.2. *If the clean event occurs, then phase 1 of* ExploreThenControlBudget *has at most*

$$\tilde{C} \cdot \log T$$

rounds, where $\tilde{C} = O\left(\frac{km^2}{\min\{\delta^2_{drift}, \sigma^2_{\min}\}\Delta^2}\right)$ *is a constant.*

3.4.3 Reduction from BwK

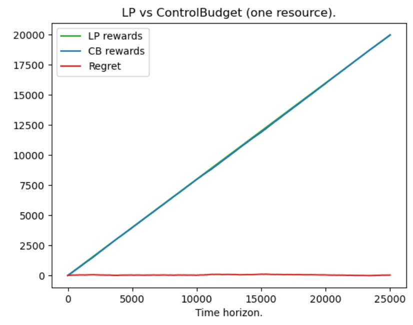
Theorem 3.4. *Suppose* $\frac{B}{T} \geq \delta_{drift}$*. Consider a* BwK *instance such that (i) for each arm and resource, the expected consumption of that resource differs from* $\frac{B}{T}$ *by at least* δ_{drift}*; and (ii) all the other assumptions required by Theorem 3.3 (Section 3.2.3) are also satisfied. Then, there is an algorithm for* BwK *whose regret satisfies the same bound as in Theorem 3.3 with the same constant* $\tilde{C}$*.*

We present the details of the reduction at the end of Section 3.9.

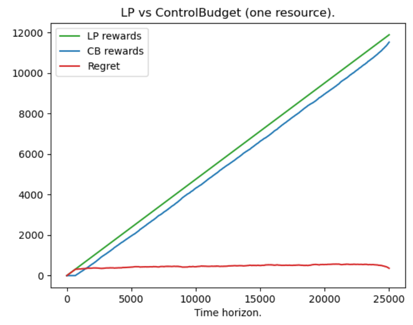
3.5 Experiments

In this section we present some simple experimental results. The implementation of our algorithms and code to reproduce the results are available at this Github repository [Kumar and Kleinberg, 2022b]. For simplicity, we only consider Bernoulli distributions, i.e., rewards are supported on $\{0, 1\}$, a positive drift arm's drifts are supported on $\{0, 1\}$, and a negative drift arm's drifts are supported on $\{0, -1\}$. We generate the data for the experiments as follows:

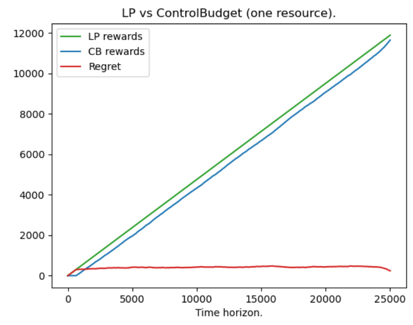
- Fig. 3.3 plot (a): We set $T = 25,000, B = 0, n = 2$ and $m = 1$. The expected reward and drifts for the arms are: $(0; 0.1), (0.8; 0.4)$. The LP solution is supported on a single positive drift arm.
- Fig. 3.3 plot (b): We set $T = 25,000, B = 400, n = 2$ and $m = 1$. The expected reward and drifts for the arms are: $(0; 0.4), (0.8; -0.3)$. The LP solution is supported on the null arm and the negative drift arm.
- Fig. 3.3 plot (c): We set $T = 25,000, B = 400, n = 3$ and $m = 1$. The expected reward and drifts for the arms are: $(0; 0.4), (0.8; -0.3), (0.1; 0.3)$. The LP solution is supported on the positive drift arm and the negative drift arm.
- Fig. 3.3 plot (d): We set $T = 25,000, B = 3, n = 3$ and $m = 2$. The expected reward and drifts for the arms are: $(0; 0.1, 0.08), (0.8; -0.2, -0.25), (0.1; 0.4, 0.5)$. The LP solution is supported on the positive drift arm and the negative drift arm.



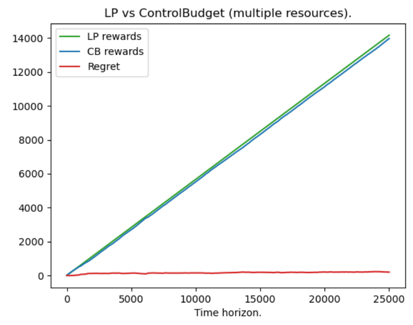
(a) ControlBudget with one resource - case 1 (positive drift arm)



(b) ControlBudget with one resource - case 2 (null plus negative drift arm)



(c) ControlBudget with one resource - case 3 (positive plus negative drift arm)



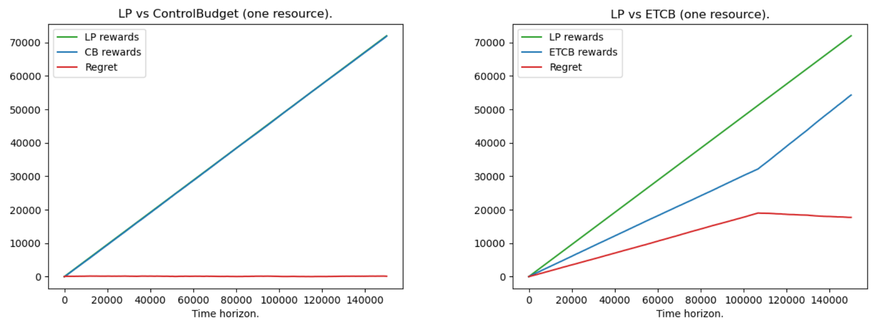
(d) ControlBudget with multiple resources

Figure 3.3: Regret of ControlBudget on a variety of test cases.

- Fig. 3.4 plot (a) and (b): We set $T = 150,000, B = 10, n = 3$ and $m = 1$. The expected reward and drifts for the arms are: $(0; 0.9), (0.8; -0.6), (0.1; 0.7)$. The LP solution is supported on the positive drift arm and the negative drift arm.

As our plots show (Fig. 3.3), our MDP policy, ControlBudget, performs quite well and achieves constant regret.

Our learning algorithm does not perform as well empirically due to large con-



(a) ControlBudget (b) ExploreThenControlBudget

Figure 3.4: Regret of ControlBudget and ExploreThenControlBudget on the same test case. We modify ExploreThenControlBudget to use the empirical means instead of UCB/LCB estimates for phase one as described in Section 3.5.

stant factors. Specifically, the number of rounds required for the confidence radius to be small enough for phase one to successfully identify X^* and J^* is too large. In our simple test cases, if we simply consider the empirical means, which are very close to the true means, instead of the UCB/LCB estimates for phase one, then the learning algorithm performs as expected: it achieves logarithmic regret by spending a logarithmic number of rounds identifying X^* and J^*, and achieves constant regret thereafter (Fig. 3.4).

3.6 Discussion

In this chapter we introduced a natural generalization of BwK that allows non-monotonic resource utilization. We first considered the setting when the decision-maker knows the true distributions and presented an MDP policy with *constant* regret against an LP relaxation. Then we considered the setting when the decision-maker does not know the true distributions and presented a learning algorithm with *logarithmic* regret against the same LP relaxation. Finally, we also presented a reduction from BwK to our model and showed a regret bound that matches existing results [Li et al., 2021a].

An important direction for future research is to obtain optimal regret bounds. The regret bound for our algorithm scales as $O(\log(T)\text{poly}(k)\text{poly}(m)\Delta^{-2})$, where k is the number of arms, m is the number of resources and Δ is the suboptimality parameter. A modification to our algorithm along the lines of Flajolet and Jaillet [2015, algorithm UCB-Simplex] that considers each support set of the LP solution explicitly leads to a regret bound that scales as $O(\log(T)\text{poly}(k)2^m\Delta^{-1})$. It is an open question, even for BwK, to obtain a regret bound that scales as $O(\log(T)\text{poly}(k)\text{poly}(m)\Delta^{-1})$ or show that the trade-off between the dependence on the number of resources and the suboptimality parameter is unavoidable.

Another natural follow-up to our work is to develop further extensions, such as considering an infinte set of arms [Kleinberg et al., 2019], studying adversarial observations [Auer et al., 2002b, Immorlica et al., 2022], or incorporating contextual information [Slivkins, 2014, Agrawal and Devanur, 2016] as has been the case elsewhere throughout the literature on bandits.

3.7 Proofs for Section 3.3.1: MDP Policy, One Resource

Lemma 3.3.2. *If the LP solution is supported on a positive drift arm x^p, then*

$$\mathsf{Reg}_T^{\mathsf{BwK}}(\mathsf{ControlBudget}) \le \tilde{C},$$

where $\tilde{C} = O\left(\delta_{drift}^{-3} \ln\left(\left(1 - \exp\left(-\frac{\delta_{drift}^2}{8}\right)\right)^{-1}\right)\right)$ *is a constant.*

Proof. When the LP solution is supported on a positive drift arm x^p, $\mathsf{OPT}_{\mathsf{LP}} = \mu_{x^p}^r$ because the LP plays it with probability 1. Therefore, the regret is equal to the expected number of times $\mathsf{ControlBudget}$ (Algorithm 1) pulls the null arm. This, in turn, is equal to the expected number of rounds in which the budget is less than 1.

Define

$$b_0 = 8\delta_{\text{drift}}^{-2} \ln\left(\frac{2}{1 - \exp\left(-\frac{\delta_{\text{drift}}^2}{8}\right)}\right).$$

Then, we have that for all $b \ge b_0$,

$$\begin{aligned}
\sum_{k=b}^{\infty} \mathbf{Pr}\left[B_{s+k} \in [0,1) \mid B_s \in [b, b+1)\right] &\le \sum_{k=b}^{\infty} \exp\left(-\frac{\delta_{\text{drift}}^2 k}{8}\right) \\
&= \exp\left(-\frac{\delta_{\text{drift}}^2 b}{8}\right)\left(1 - \exp\left(-\frac{\delta_{\text{drift}}^2}{8}\right)\right)^{-1}.
\end{aligned}$$

where the first inequality follows from Azuma-Hoeffding's inequality. By our choice of b_0, we have that

$$\sum_{k=b}^{\infty} \mathbf{Pr}\left[B_{s+k} \in [0,1) \mid B_s \in [b, b+1)\right] \le \frac{1}{2}. \tag{3.6}$$

In words, the probability that the budget ever drops below 1 once it exceeds b_0 is at most $\frac{1}{2}$. Now, consider the following recursive definition for two disjoint sequence of indices s_i and s'_i. Let $s_0 = \min\{t \geq 1 : B_{t-1} \in [0, 1)\}$, and define

$$\begin{aligned} s'_i &= \min\{t > s_i : B_{t-1} \geq b_0 \text{ or } t - 1 = T\} \\ s_{i+1} &= \min\{t > s'_i : B_{t-1} \in [0, 1)\}. \end{aligned}$$

In words, s'_i denotes the first round after s_i in which the budget is at least b_0 and s_{i+1} denotes the first round after s'_i in which the budget is less than 1. Note that Eq. (3.6) implies that

$$\mathbf{Pr}\left[s_i \text{ is defined } \mid s'_{i-1} \text{ is defined}\right] \leq \frac{1}{2}.$$

Therefore,

$$\mathbf{Pr}\left[s_i \text{ is defined}\right] \leq \prod_{j=1}^{i} \mathbf{Pr}\left[s_j \text{ is defined } \mid s'_{j-1} \text{ is defined}\right] \leq \frac{1}{2^i}.$$

Now, we can upper bound the expected number of rounds in which the budget is below 1 as

$$\begin{aligned}
\mathbb{E}\left[\sum_{t=1}^{T} \mathbb{1}[B_{t-1} < 1]\right] &= \sum_{i=0}^{T-1} \mathbf{Pr}\left[s_i \text{ is defined}\right] \mathbb{E}\left[\sum_{t=s_i}^{s_i'} \mathbb{1}[B_{t-1} < 1] \mid s_t \text{ is defined}\right] \\
&\le \sum_{t=0}^{T-1} 2^{-i} \mathbb{E}\left[\sum_{t=s_i}^{s_i'} \mathbb{1}[B_{t-1} < 1] \mid s_t \text{ is defined}\right] \\
&\le \sum_{t=0}^{T-1} 2^{-i} \mathbb{E}\left[s_i' - s_i\right] \\
&\le \sum_{t=0}^{T-1} 2^{-i} \frac{1}{\delta_{\text{drift}}} \mathbb{E}\left[B_{s_i'} - B_{s_i}\right] \qquad (3.7) \\
&\le \sum_{t=0}^{T-1} 2^{-i} \frac{1}{\delta_{\text{drift}}} (b_0 + 1) \\
&\le 2 \frac{b_0 + 1}{\delta_{\text{drift}}},
\end{aligned}$$

where Eq. (3.7) follows because both the null arm and the positive drift arm have drift at least δ_{drift}. Therefore, we have that

$$\mathbb{E}\left[\sum_{t=1}^{T} \mathbb{1}[B_{t-1} < 1]\right] \le \tilde{C},$$

where

$$\tilde{C} = O\left(\delta_{\text{drift}}^{-3} \ln\left(\frac{2}{1 - \exp\left(-\frac{\delta_{\text{drift}}^2}{8}\right)}\right)\right). \qquad (3.8)$$

This completes the proof. ■

Lemma 3.3.3. *If the LP solution is supported on the null arm x^0 and a negative drift arm x^n, then*

$$\mathsf{Reg}_T^{\mathsf{BwK}}(\mathsf{ControlBudget}) \le \tilde{C} \cdot \mathbb{E}[B_T],$$

where $\tilde{C} = O(\delta_{drift}^{-1})$ is a constant.

Proof. Let p^* denote the optimal solution to the LP relaxation and note that Tp^*_x denotes the expected number of times the LP plays arm x. Since the LP solution is supported on two arms, both the budget and sum-to-one constraints are tight. Therefore, we have

$$D(Tp^*) = b_{\mathsf{LP}}, \tag{3.9}$$

where

$$D = \begin{bmatrix} \mu^d_{x^0} & \mu^d_{x^n} \\ 1 & 1 \end{bmatrix}, \; p^* = \begin{bmatrix} p^*_{x^0} \\ p^*_{x^n} \end{bmatrix}, \; b_{\mathsf{LP}} = \begin{bmatrix} -B \\ T \end{bmatrix}. \tag{3.10}$$

Let N_x denote the number of times ControlBudget (Algorithm 1) plays arm x. Since it plays the null arm x^0 and the negative drift arm x^n, the sum-to-one constraint is tight. However, the budget constraint may not be tight because there may be leftover budget. Therefore, we have

$$DN = b_{\mathsf{LP}} - b, \tag{3.11}$$

where

$$N = \begin{bmatrix} \mathbb{E}[N_{x^0}] \\ \mathbb{E}[N_{x^n}] \end{bmatrix}, \; b = \begin{bmatrix} -E[B_T] \\ 0 \end{bmatrix}. \tag{3.12}$$

Define

$$\xi = \begin{bmatrix} \xi_{x^0} \\ \xi_{x^n} \end{bmatrix} = \begin{bmatrix} Tp^*_{x^0} - \mathbb{E}[N_{x^0}] \\ Tp^*_{x^n} - \mathbb{E}[N_{x^n}] \end{bmatrix}.$$

Subtracting Eq. (3.11) from Eq. (3.9) we have $\xi = D^{-1}b$, where the LP constraint matrix D is invertible by our assumption that the drifts are nonzero. Finally, letting

μ^r denote the vector of expected rewards, the regret can be expressed as

$$\begin{aligned}
\mathrm{Reg}_T^{\mathsf{BwK}}(\mathsf{ControlBudget}) &= \xi^T \mu^r \\
&\leq |\xi^T \mu^r| \\
&\leq \|\xi\|_1 \|\mu^r\|_\infty \\
&\leq \|D^{-1}\|_1 \|b\|_1 \\
&\leq C_{\delta_{\text{drift}}} \mathbb{E}[B_T],
\end{aligned}$$

where $C_{\delta_{\text{drift}}} = O(\delta_{\text{drift}}^{-1})$ is a constant. This completes the proof. ■

Lemma 3.3.4. *If the LP solution is supported on a positive drift arm x^p and a negative drift arm x^n, then*

$$\mathrm{Reg}_T^{\mathsf{BwK}}(\mathsf{ControlBudget}) \leq \tilde{C} \cdot \max\{\mathbb{E}[B_T], \mathbb{E}[N_{x^0}]\},$$

where $\mathbb{E}[N_{x^0}]$ denotes the expected number null arm pulls and $\tilde{C} = O(\delta_{drift}^{-1})$ is a constant.

Proof. Let p^* denote the optimal solution to the LP relaxation and note that Tp_x^* denotes the expected number of times the LP plays arm x. Since the LP solution is supported on two arms, both the budget and sum-to-one constraints are tight. Therefore, we have

$$D(Tp^*) = b_{\mathsf{LP}}, \tag{3.13}$$

where

$$D = \begin{bmatrix} \mu_{x^p}^d & \mu_{x^n}^d \\ 1 & 1 \end{bmatrix}, \; p^* = \begin{bmatrix} p_{x^p}^* \\ p_{x^n}^* \end{bmatrix}, \; b_{\mathsf{LP}} = \begin{bmatrix} -B \\ T \end{bmatrix}. \tag{3.14}$$

Let N_x denote the number of times ControlBudget (Algorithm 1) plays arm x. Since it plays the null arm x^0 when the budget is less than 1 and may have leftover budget, neither the budget nor the sum-to-one constraint are tight. Therefore, we have

$$DN = b_{\mathsf{LP}} - b, \tag{3.15}$$

where

$$N = \begin{bmatrix} \mathbb{E}[N_{x^p}] \\ \mathbb{E}[N_{x^n}] \end{bmatrix}, \; b = \begin{bmatrix} -E[B_T] \\ \mathbb{E}[N_{x^0}] \end{bmatrix}. \tag{3.16}$$

Define

$$\xi = \begin{bmatrix} \xi_{x^p} \\ \xi_{x^n} \end{bmatrix} = \begin{bmatrix} Tp^*_{x^p} - \mathbb{E}[N_{x^p}] \\ Tp^*_{x^n} - \mathbb{E}[N_{x^n}] \end{bmatrix}.$$

Subtracting Eq. (3.15) from Eq. (3.13) we have $\xi = D^{-1}b$, where the LP constraint matrix D is invertible by our assumption that the drifts are nonzero. Finally, letting μ^r denote the vector of expected rewards, the regret can be expressed as

$$\begin{aligned} \mathsf{Reg}_T^{\mathsf{BwK}}(\mathsf{ControlBudget}) &= \xi^T \mu^r \\ &\le |\xi^T \mu^r| \\ &\le \|\xi\|_1 \|\mu^r\|_\infty \\ &\le \|D^{-1}\|_1 \|b\|_1 \\ &\le C_{\delta_{\text{drift}}} \left(\mathbb{E}[B_T] + \mathbb{E}[N_{x^0}]\right), \end{aligned}$$

where $C_{\delta_{\text{drift}}} = O(\delta_{\text{drift}}^{-1})$ is a constant. This completes the proof. ■

Lemma 3.3.5. *If the LP solution is supported on a positive drift arm x^p and a negative* *drift arm x^n, and $c \geq \frac{12}{\delta^2_{drift}}$, then*

$$\mathbb{E}[N_{x^0}] \leq \tilde{C},$$

where $\tilde{C} = O\left(\delta^{-3}_{drift} \ln\left(\left((1 - \exp\left(-\frac{\delta^2_{drift}}{8}\right)\right)^{-1}\right)\right)$ is a constant.

Proof. Divide the T rounds into two phases: $P_1 = \{1, \ldots, T - \exp(3/c)\}$ and $P_2 = \{1, \ldots, T\} \setminus P_1$. Note that P_2 consists of $\exp(3/c) = O(\exp(\delta^2_{\text{drift}})) = O(1)$ rounds, where the last equality follows because drifts are bounded by 1. Therefore, the expected number of null arm pulls in this phase is $O(1)$ and it suffices to bound the expected number of null arm pulls in P_1.

Consider the following recursive definition for three disjoint sequences of indices t_i, t'_i and t''_i. Let $t_0 = 0$, and define

$$t'_i = \min\{t > t_i : B_{t-1} \geq \tau_t \text{ or } t - 1 = T\},$$

$$t''_i = \min\{t > t'_i : B_{t-1} < \tau_t\},$$

$$t_{i+1} = \min\{t > t''_i : B_t < 1\}.$$

We can bound the expected number of rounds in which the budget is less than

1 as

$$
\begin{aligned}
&\mathbb{E}\left[\sum_{t=1}^{T}\mathbb{1}[B_{t-1}<1]\right]\\
&=\sum_{i=0}^{T-1}\mathbf{Pr}\left[t_i \text{ exists }\right]\mathbb{E}\left[\sum_{t=t_i}^{t_i'}\mathbb{1}[B_{t-1}<1]\mid t_i \text{ exists}\right]\\
&\le \underbrace{\mathbb{E}\left[\sum_{t=t_0}^{t_0'-1}\mathbb{1}[B_{t-1}<1]\mid t_0 \text{ exists}\right]}_{(a)}\\
&\qquad+\sum_{i=0}^{T-1}\mathbf{Pr}\left[t_{i+1} \text{ exists } \mid t_i', t_i'' \text{ exist}\right]\underbrace{\mathbb{E}\left[\sum_{t=t_i}^{t_i'-1}\mathbb{1}[B_{t-1}<1]\mid t_i \text{ exists}\right]}_{(a)}.
\end{aligned}
$$

In rounds $\{t_i,\dots,t_i'-1\}$, the algorithm pulls the null and positive drift arms. The proof of Lemma 3.3.2 shows that the expected number of null arm pulls in these rounds is at most $\tilde{C}$, where $\tilde{C}$ is defined in Eq. (3.8). Therefore, we can bound the term (a) in the above inequality by $\tilde{C}$ and we have that

$$
\mathbb{E}\left[\sum_{t=1}^{T}\mathbb{1}[B_{t-1}<1]\right]\le\tilde{C}\left(1+\sum_{i=0}^{T-1}\mathbf{Pr}\left[t_{i+1} \text{ exists } \mid t_i', t_i'' \text{ exist}\right]\right). \tag{3.17}
$$

If t_i' exists, then $B_{t_i'-1}\ge\tau_{t_i'}$. If t_i'' exists, then $\tau_{t_i''}-1\le B_{t_i''-1}<\tau_{t_i''}$ because (i) t_i'' is the first round after t_i' in which the budget is below the threshold; and (ii) the drifts are bounded by 1, so it cannot be lower than $\tau_{t_i''}-1$. The algorithm pulls the negative drift arm x^n in the rounds $\{t_i',\dots,t_i''-1\}$ and the positive drift arm x^p in the rounds $\{t_i'',\dots,t_{i+1}-1\}$. Since the drifts are bounded by 1, it takes at least $\tau_{t_i''}-2$ rounds for the budget to drop below 1 after repeated pulls of x^p. Using this and the observation that the budget dropping below 1 is contained in the event that the total drift in those rounds is nonpositive, we can bound (a) as

$$\begin{aligned}
\mathbf{Pr}\left[t_{i+1} \text{ exists } \mid t_i'', t_i' \text{ exist}\right] &\leq \sum_{q=t_i''+\tau_{t_i''}-2}^{T} \mathbf{Pr}\left[\sum_{t=t_i''+1}^{q} d_t \leq 0\right] \\
&\leq \sum_{q=t_i''+\tau_{t_i''}-2}^{T} \exp\left(-\frac{1}{4}\delta_{\text{drift}}^2(\tau_{t_i''}-2)\right) \\
&\leq \sum_{q=t_i''+\tau_{t_i''}-2}^{T} \exp\left(-\frac{1}{4}\delta_{\text{drift}}^2\tau_{t_i''}\right) \\
&= \sum_{q=t_i''+\tau_{t_i''}-2}^{T} \exp\left(-\frac{1}{4}\delta_{\text{drift}}^2 c\log(T-t_i'')\right) \\
&\leq \sum_{q=t_i''+\tau_{t_i''}-2}^{T} (T-t_i'')^{-3},
\end{aligned}$$

where the second inequality follows from the Azuma-Hoeffding inequality applied to the sequence of drifts sampled from x^p and the last inequality follows because $c \geq \frac{12}{\delta_{\text{drift}}^2}$. The summation is over at most $T - t_i''$ terms because there are at most $T - t_i''$ rounds left after round t_i''. Therefore, we have that

$$\mathbf{Pr}\left[t_{i+1} \text{ exists } \mid t_i'', t_i' \text{ exist}\right] \leq (T-t_i'')^{-2}.$$

Substituting this in Eq. (3.17), we have that

$$\begin{aligned}
\mathbb{E}\left[\sum_{t=1}^{T} \mathbb{1}[B_{t-1} < 1]\right] &\leq \tilde{C}\left(1 + \sum_{i=0}^{T-1} \mathbf{Pr}\left[t_{i+1} \text{ exists } \mid t_i', t_i'' \text{ exist}\right]\right) \\
&\leq \tilde{C}\left(1 + \sum_{i=0}^{T-1}(T-t_i'')^{-2}\right) \\
&\leq \tilde{C}\left(1 + \sum_{i=0}^{\infty}(T-t_i'')^{-2}\right) \\
&\leq \tilde{C}\left(1 + \frac{\pi^2}{6}\right).
\end{aligned}$$

This completes the proof. ■

Lemma 3.3.6. *If the LP solution is supported on two arms, and* $c \geq \frac{12}{\delta_{drift}^2}$*, then*

$$\mathbb{E}[B_T] \leq \tilde{C},$$

where $\tilde{C} = \tilde{O}\left(\left(1 - \exp\left(\delta_{drift}^2\right)\right)^{-2} + \delta_{drift}^{-2}\right)$ *is a constant.*

Proof. Let E_q denote the event that the negative drift arm x^n is pulled consecutively in exactly the last q rounds, i.e., $x_t = x^n$ for all $t \geq T - q + 1$ and $x_t \in \{x^0, x^p\}$ for $t = T - q$ (if $q \neq T$). Note that the events $(E_q : q = 0, \dots, T)$ are disjoint. Let S_q denote the event that the total drift in the last q pulls of x^n is greater than $\frac{1}{2}\mu_{x^n}^d q$, i.e., $\sum_{t \geq T-q+1} d_t > \frac{1}{2}\mu_{x^n}^d q$. We can upper bound the expected leftover budget by conditioning on these events as follows.

$$\begin{aligned}
\mathbb{E}[B_T] &= \sum_{q=0}^{T} \mathbf{Pr}[E_q]\, \mathbb{E}[B_T \mid E_q] \\
&\leq \sum_{q=0}^{T} \mathbb{E}[B_T \mid E_q] \\
&= \sum_{q=0}^{T} \underbrace{\mathbb{E}[B_T \mid E_q, S_q]}_{(a)}\ \underbrace{\mathbf{Pr}[S_q \mid E_q]}_{(b)} + \underbrace{\mathbb{E}[B_T \mid E_q, S_q^c]}_{(c)}\ \underbrace{\mathbf{Pr}[S_q^c \mid E_q]}_{(d)}.
\end{aligned}$$

If $q = 0$, then the expected leftover budget is trivially at most a constant. We can bound the four terms for $q \geq 1$ as follows:

(a) We have

$$\mathbb{E}[B_T \mid E_q, S_q] \leq c \log q + q$$

because (i) ControlBudget (Algorithm 1) pulls x^0 or x^p in round $T-q$ if $B_{T-q-1} < \tau_{T-q} = c\log q$; and (ii) conditioned on the event S_q, the total drift in the last q rounds can be at most q as the drifts are bounded by 1.

(b) We have

$$\mathbf{Pr}[S_q \mid E_q] \leq \exp\left(-\frac{1}{16}(\mu^d_{x^n})^2 q\right)$$

because (i) the sequence of drifts observed from q pulls of the negative drift arm x^n is a supermartingale difference sequence; and (ii) by the Azuma-Hoeffding inequality, the probability the sum S_q is greater than half its expected value is at most $\exp\left(-\frac{1}{16}(\mu^d_{x^n})^2 q\right)$.

(c) We have

$$\mathbb{E}[B_T \mid E_q, S^c_q] \leq \left(c\log q + \frac{1}{2}\mu^d_{x^n} q\right)$$

because (i) ControlBudget (Algorithm 1) pulls x^0 or x^p in round $T-q$ if $B_{T-q-1} < \tau_{T-q} = c\log q$; and (ii) conditioned on the event S^c_q, the total drift in the last q rounds can be at most $\frac{1}{2}\mu^d_{x^n} q$.

(d) We have

$$\mathbf{Pr}[S^c_q \mid E_q] \leq 1$$

trivially.

Therefore,

$$\mathbb{E}[B_T] \leq \sum_{q=0}^{T} \underbrace{(c\log q + q)\exp\left(-\frac{1}{16}(\mu^d_{x^n})^2 q\right)}_{(e)} + \underbrace{\left(c\log q + \frac{1}{2}\mu^d_{x^n} q\right)}_{(f)}.$$

This summation is a constant in terms of T:

1. Term (e) is a constant because $c \log q < q$ for q large enough and $\sum_{q=1}^{\infty} q \exp(-aq)$ converges to $\exp(a)(1 - \exp(a))^{-2}$.
2. Term (f) is a constant because this term is negative for q large enough as $\mu^d_{x^n} < 0$ and is maximized at $q = \frac{2c}{|\mu^d_{x^n}|}$.

Finally, we can bound the expected leftover budget as

$$\mathbb{E}[B_T] \le \tilde{C} = \tilde{O}\left(\left(1 - \exp\left(\frac{\delta^2_{\text{drift}}}{16}\right)\right)^{-2} + \frac{1}{\delta^2_{\text{drift}}}\right),$$

where the last equality follows when $c \ge \frac{6}{\delta^2_{\text{drift}}}$. This completes the proof. ∎

3.8 Proofs for Section 3.3.2: MDP Policy, Multiple Resources

Lemma 3.3.9. *[Flajolet and Jaillet, 2015, Lemma 14] In each round t, $\gamma_t \ge \gamma^* = \frac{\sigma_{\min} \min\{\delta_{support}, \delta_{slack}\}}{4m}$.*

Proof. It suffices to show that $\gamma = \frac{\sigma_{\min} \min\{\delta_{\text{support}}, \delta_{\text{slack}}\}}{4m}$ is a feasible solution the Eq. (3.2).

First, we show that $p = D^{-1}(b + \gamma s_t) \ge 0$. For each $x \in X^*$,

$$\begin{aligned} e_x^T D^{-1}(b + \gamma s_t) &= e_x^T D^{-1} b + \gamma e_x^T D^{-1} s_t \\ &= p^*_x + \gamma e_x^T D^{-1} s_t \\ &\ge \delta_{\text{support}} - \gamma \|D^{-1} s_t\|_2 \\ &\ge \delta_{\text{support}} - \gamma \frac{1}{\sigma_{\min}} \sqrt{m} \\ &\ge 0. \end{aligned}$$

Second, we show that for any non-binding resource j, $(\mu^{d,j})^T D^{-1}(b + \gamma s_t) \geq \frac{\delta_{\text{slack}}}{2}$:

$$
\begin{aligned}
(\mu^{d,j})^T D^{-1}(b + \gamma s_t) &= \sum_{x \in X} \mu_x^{d,j} p_x^* + \gamma (\mu^{d,j})^T D^{-1} s_t \\
&\geq \delta_{\text{slack}} - \gamma |(\mu^{d,j})^T D^{-1} s_t| \\
&\geq \delta_{\text{slack}} - \gamma ||(\mu^{d,j})^T||_2 ||D^{-1}||_2 ||s_t||_2 \\
&\geq \delta_{\text{slack}} - \gamma \frac{1}{\sigma_{\min}} m \\
&\geq \frac{\delta_{\text{slack}}}{2} \\
&\geq \frac{\gamma}{2},
\end{aligned}
$$

where the last inequality follows because $\sigma_{\min}, \delta_{\text{slack}}, \delta_{\text{support}} < 1$. ■

Lemma 3.3.10. *If the LP solution does not include the null arm in its support, then*

$$\mathbb{E}[N_{x^0}] \leq \tilde{C},$$

where $\tilde{C} = O\left(m(\gamma^*)^{-3} \ln\left(\left((1 - \exp\left(-\frac{\gamma^{*2}}{8}\right)\right)^{-1}\right)\right)$ *is a constant.*

Proof. Divide the T rounds into two phases: $P_1 = \{1, \ldots, T - \exp(3/c)\}$ and $P_2 = \{1, \ldots, T\} \setminus P_1$. Note that P_2 consists of $\exp(3/c) = O(\exp((\gamma^*)^2)) = O(1)$ rounds, where the last equality follows because γ^* is bounded by 1. Therefore, the expected number of null arm pulls in this phase is $O(1)$ and it suffices to bound the expected number of null arm pulls in P_1.

We can write the expected number of rounds in which there exists a resource whose budget is less than 1 as

$$\mathbb{E}\left[\sum_{t=1}^{T}\sum_{j\in\mathcal{J}}\mathbb{1}[B_{t-1,j}<1]\right]=\sum_{j\in\mathcal{J}}\underbrace{\mathbb{E}\left[\sum_{t=1}^{T}\mathbb{1}[B_{t-1,j}<1]\right]}_{(a)}.$$

We can bound term (a) above the same way as in the proof of Lemma 3.3.5 (Section 3.7) with δ_{drift} replaced by γ^*. Therefore,

$$\mathbb{E}\left[\sum_{t=1}^{T}\sum_{j\in\mathcal{J}}\mathbb{1}[B_{t-1,j}<1]\right]\leq m\tilde{C}\left(1+\frac{\pi^2}{6}\right),$$

where $\tilde{C}$ is defined in Eq. (3.8). ■

Lemma 3.3.11. *If the LP solution is supported on more than one arm, then for all $j \in J^*$*

$$\mathbb{E}[B_{T,j}]\leq\tilde{C},$$

where $\tilde{C}=\tilde{O}\left(\left(1-\exp(\gamma^{*2})\right)^{-2}+(\gamma^*)^{-2}\right)$ is a constant.

Proof. Consider an arbitrary resource $j \in J^*$. Recall the vector s_t defined in ControlBudget (Algorithm 2). If i denotes the row corresponding to resource j, then the ith entry of s_t, denoted by $s_t(i)$, is -1 if $B_{t-1,j} < \tau_t$ and $+1$ otherwise.

Let E_q denote the event that the $s_t(i)$ is equal to -1 consecutively in exactly the last q rounds, i.e., $s_t(i) = -1$ for all $t \geq T-q+1$ and $s_t(i) = +1$ for $t = T-q$ (if $q \neq T$). Note that the events $(E_q : q = 0, \ldots, T)$ are disjoint. Let S_q denote the event that the total drift for j in the last q rounds is greater than $\frac{1}{2}(-\gamma^*)q$, i.e., $\sum_{t\geq T-q+1} d_t > \frac{1}{2}(-\gamma^*)q$. We can upper bound the expected leftover budget of resource j by conditioning on these events as follows.

$$\begin{aligned}\mathbb{E}[B_{T,j}] &= \sum_{q=0}^{T} \mathbf{Pr}[E_q]\, \mathbb{E}[B_{T,j} \mid E_q] \\ &\leq \sum_{q=0}^{T} \mathbb{E}[B_{T,j} \mid E_q] \\ &= \sum_{q=0}^{T} \underbrace{\mathbb{E}[B_{T,j} \mid E_q, S_q]}_{(a)}\ \underbrace{\mathbf{Pr}[S_q \mid E_q]}_{(b)} + \underbrace{\mathbb{E}[B_{T,j} \mid E_q, S_q^c]}_{(c)}\ \underbrace{\mathbf{Pr}[S_q^c \mid E_q]}_{(d)}.\end{aligned}$$

If $q = 0$, then the expected leftover budget is trivially at most a constant. We can bound the four terms for $q \geq 1$ as follows:

(a) We have

$$\mathbb{E}[B_{T,j} \mid E_q, S_q] \leq c \log q + q$$

because (i) ControlBudget (Algorithm 2) sets $s_t(i) = +1$ in round $T - q$ if $B_{T-q-1,j} < \tau_{T-q} = c \log q$; and (ii) conditioned on the event S_q, the total drift in the last q rounds can be at most q as the drifts are bounded by 1.

(b) We have

$$\mathbf{Pr}[S_q \mid E_q] \leq \exp\left(-\frac{1}{16}(\gamma^*)^2 q\right)$$

because (i) the sequence of drifts observed in rounds $t \geq T - q + 1$ is a supermartingale difference sequence with $\mathbb{E}[d_{s,j} \mid d_{T-q+1,j}, \ldots, d_{s-1,j}] \leq -\gamma^*$; and (ii) by the Azuma-Hoeffding inequality, the probability the sum S_q is greater than half its expected value is at most $\exp\left(-\frac{1}{16}(\gamma^*)^2 q\right)$.

(c) We have

$$\mathbb{E}[B_{T,j} \mid E_q, S_q^c] \leq \left(c \log q + \frac{1}{2}(-\gamma^*) q\right)$$

because (i) ControlBudget (Algorithm 2) sets $s_t(i) = +1$ in round $T - q$ if $B_{T-q-1,j} < \tau_{T-q} = c \log q$; and (ii) conditioned on the event S_q^c, the total drift in the last q rounds can be at most $\frac{1}{2}(-\gamma^*)q$.

(d) We have

$$\mathbf{Pr}[S_q^c \mid E_q] \leq 1$$

trivially.

Therefore,

$$\mathbb{E}[B_{T,j}] \leq \sum_{q=0}^{T} \underbrace{(c \log q + q) \exp\left(-\frac{1}{16}(\gamma^*)^2 q\right)}_{(e)} + \underbrace{\left(c \log q + \frac{1}{2}(-\gamma^*)q\right)}_{(f)}.$$

This summation is a constant in terms of T:

1. Term (e) is a constant because $c \log q < q$ for q large enough and $\sum_{q=1}^{\infty} q \exp(-aq)$ converges to $\exp(a)(1 - \exp(a))^{-2}$.
2. Term (f) is a constant because this term is negative for q large enough and is maximized at $q = \frac{2c}{\gamma^*}$.

Finally, we can bound the expected leftover budget as

$$\mathbb{E}[B_{T,j}] \leq \tilde{C} = \tilde{O}\left(\left(1 - \exp\left(\frac{\gamma^{*2}}{16}\right)\right)^{-2} + \frac{1}{\gamma^{*2}}\right),$$

where the last equality follows when $c \geq \frac{6}{\gamma^{*2}}$. This completes the proof. ■

3.9 Proofs for Section 3.4: Learning Algorithm

Lemma 3.4.1. *The clean event occurs with probability at least* $1 - 5kmT^{-2}$.

Proof. It suffices to show that the complement of the clean event occurs with probability at most $5kmT^{-2}$.

For (i) in the definition of the clean event (Definition 3.4.2), first fix an arm $x \in \mathcal{X}$ and a round $t \geq k$. By taking a union bound over the components of the outcome vector and using the Azuma-Hoeffding inequality, we have

$$\begin{aligned}
\mathbf{Pr}\left[\mu_x^o \notin [\mathsf{LCB}_t(x), \mathsf{UCB}_t(x)]\right] &\leq 2(m+1)\exp\left(-2n_t(x)\mathsf{rad}_t(x)^2\right) \\
&\leq 4m\exp\left(-2n_t(x)\frac{8\log T}{n_t(x)}\right) \\
&\leq 4mT^{-16}.
\end{aligned}$$

Now, by taking a union bound over k arms in $\mathcal{X}$ and all rounds $t \geq k$, we have

$$\mathbf{Pr}\left[\exists x \in \mathcal{X},\ \exists t \geq k \text{ s.t. } \mu_x^o \notin [\mathsf{LCB}_t(x), \mathsf{UCB}_t(x)]\right] \leq 4kmT^{-15} \leq 4kmT^{-2}.$$

For (ii) in the definition of the clean event (Definition 3.4.2), a similar approach works. Let $S_{n,j}$ denote the sum of the drifts for resource $j \in \mathcal{J}$ after n pulls of the null arm x^0. By the union bound and Azuma-Hoeffding inequality,

$$\begin{aligned}
\mathbf{Pr}\left[\exists j \in \mathcal{J} \text{ s.t. } S_{n,j} < w\right] &\le \sum_{j\in\mathcal{J}} \exp\left(-\frac{1}{4} w \mu_{x^0}^{d,j}\right) \\
&\le m \exp\left(-\frac{1}{4} w \delta_{\mathrm{drift}}\right) \\
&\le m \exp\left(-\frac{1}{4} \frac{1024 k m^2 \log T}{\delta_{\mathrm{drift}}^2 \sigma_{\min}^2} \delta_{\mathrm{drift}}\right) \\
&= m \exp\left(-\frac{256 k m^2 \log T}{\delta_{\mathrm{drift}} \sigma_{\min}^2}\right) \\
&\le m \exp\left(-256 k m^2 \log T\right) \qquad (3.18) \\
&\le m \exp\left(-256 \log T\right) \\
&\le m T^{-2},
\end{aligned}$$

where Eq. (3.18) follows because $\delta_{\mathrm{drift}} \in (0, 1]$ and $\sigma_{\min} \in (0, 1)$. This shows that the probability of the complement of the clean event is at most $4kmT^{-2} + mT^{-2} \le 5kmT^{-2}$ and completes the proof. ■

Lemma 3.4.2. *If the clean event occurs, then* $\mathsf{UCB}_t(\mathsf{OPT}_{\mathsf{LP}}) - \mathsf{LCB}_t(\mathsf{OPT}_{\mathsf{LP}}) \le \frac{8m}{\sigma_{\min}} \mathsf{rad}_t$*. A similar statement is true for* OPT_{-x} *and* OPT_{-j} *for all* $x \in \mathcal{X}$ *and* $j \in \mathcal{J}$*.*

Proof. We will prove the lemma for $\mathsf{OPT}_{\mathsf{LP}}$ because the other cases are similar. Simplifying and overloading notation for this proof, we denote the probability simplex over k dimensions as Δ_k, and the vector of expected rewards, the matrix of expected drifts and the right-hand side of the budget constraints as

$$r = \begin{bmatrix} \mu_1^r \\ \vdots \\ \mu_k^r \end{bmatrix}, \quad D = \begin{bmatrix} \mu_1^{d,1} & \dots & \mu_k^{d,1} \\ & \ddots & \\ \mu_1^{d,m} & \dots & \mu_k^{d,m} \end{bmatrix}, \quad b = -\frac{B}{T}\mathbf{1}.$$

We will use $\bar{r}$ and $\bar{D}$ to denote the empirical versions of the rewards and drifts. We will also use adding a scalar to a vector or matrix to denote adding the scalar to the vector or matrix component-wise. (For example, $\bar{r} + \mathsf{rad}_t$ denotes adding the scalar rad_t to the vector of empirical rewards.) Now, we can write

$$\begin{aligned}
\mathsf{OPT}_{\mathsf{LP}} &= \max_{p \in \Delta_k} r^T p && \text{s.t. } Dp \geq b, && (3.19)\\
\mathsf{UCB}_t(\mathsf{OPT}_{\mathsf{LP}}) &= \max_{q \in \Delta_k} (\bar{r} + \mathsf{rad}_t)^T q && \text{s.t. } (\bar{D} + \mathsf{rad}_t) q \geq b \\
&\leq \max_{q \in \Delta_k} (r + 2\mathsf{rad}_t)^T q && \text{s.t. } (D + 2\mathsf{rad}_t) q \geq b \\
&\leq 2\mathsf{rad}_t + \max_{q \in \Delta_k} r^T q && \text{s.t. } Dq \geq b - 2\mathsf{rad}_t, && (3.20)
\end{aligned}$$

where the second-last inequality follows because we are conditioning on the clean event. Therefore, using D' and b' to denote the submatrix and subvector corresponding to the binding constraints, p^* to denote the optimal solution of the LP in Eq. (3.19), and q^* to denote the optimal solution of the LP in Eq. (3.20), we have

$$\begin{aligned}
\mathsf{UCB}_t(\mathsf{OPT}_{\mathsf{LP}}) - \mathsf{OPT}_{\mathsf{LP}} &\leq 2\mathsf{rad}_t + |r^T p^* - r^T q^*| \\
&\leq 2\mathsf{rad}_t + |r^T (D')^{-1} b' - r^T (D')^{-1} (b' - 2\mathsf{rad}_t)| \\
&\leq 2\mathsf{rad}_t + \|r\|_2 \|(D')^{-1}\|_2 \|2\mathsf{rad}_t\|_2 \\
&\leq 2\mathsf{rad}_t + 2m\mathsf{rad}_t \frac{1}{\sigma_{\min}} \\
&\leq \frac{4m}{\sigma_{\min}} \mathsf{rad}_t,
\end{aligned}$$

where last inequality follows because $\sigma_{\min} < 1 \leq m$. Since the LCB is defined by subtracting rad_t from the empirical means, we obtain the same upper bound on $\mathsf{OPT}_{\mathsf{LP}} - \mathsf{LCB}_t(\mathsf{OPT}_{\mathsf{LP}})$ and using the triangle inequailty completes the proof. ■

Theorem 3.3. *If $c \geq \frac{12}{\gamma^*}$, the regret of* ExploreThenControlBudget *(Algorithm 3) satisfies*

$$\mathsf{Reg}_T^{\mathsf{BwK}}(\mathsf{ExploreThenControlBudget}) \leq \tilde{C} \cdot \log T,$$

where γ^* (defined in Lemma 3.3.9), $\tilde{C}'$ and $\tilde{C}$ are constants with $\tilde{C}'$ denoting the constant in Theorem 3.2 and

$$\tilde{C} = O\left(\frac{km^2}{\min\{\delta_{drift}^2, \sigma_{\min}^2\}\Delta^2} + k(\gamma^*)^{-2} + \tilde{C}'\right).$$

Proof. Since the complement of the clean event occurs with probability at most $O(kmT^{-2})$ and contributes $O(kmT^{-1})$ to the regret, it suffices to bound the regret conditioned on the clean event. So, condition on the clean event for the rest of the proof. Phase one contributes at most

$$O\left(\frac{km^2}{\min\{\delta_{\text{drift}}^2, \sigma_{\min}^2\}\Delta^2}\right) \cdot \log T$$

to the regret by Corollary 3.4.2. Phase two contributes at most

$$O\left(\frac{k}{\gamma^{*2}}\right) \log T$$

to the regret.

Observe that after phase two, $\mathsf{rad}_t(x) \leq \frac{\gamma^{*2}}{2}$ for all $x \in X^*$. Combining this with Eqs. (3.2) to (3.5), we have that $(\gamma^*, D^{-1}(b+\gamma^* s_t))$ is a feasible solution to the optimization problem solved by ExploreThenControlBudget (Algorithm 3). Therefore, (γ_t, p_t) ensure that there is drift of magnitude at least $\frac{\gamma^*}{8}$ in the “correct directions”. As noted in the end of Section 3.3.2, the regret analysis of ControlBudget (Algorithm 2) requires the algorithm to know X^*, J^*, and find a probability vector p_t

that ensures drifts bounded away from zero in the "correct directions". Therefore, by Theorem 3.2, phase three contributes at most $\tilde{C}'$ to the regret, where $\tilde{C}$ is the constant in Theorem 3.2. Combining the contribution from the three phases, we have that

$$\mathsf{Reg}_T^{\mathsf{BwK}}(\mathsf{ExploreThenControlBudget}) \leq \tilde{C} \cdot \log T,$$

where γ^* (defined in Lemma 3.3.9) and $\tilde{C}$ are constants with

$$\tilde{C} = O\left(\frac{km^2}{\min\{\delta_{\text{drift}}^2, \sigma_{\min}^2\}\Delta^2} + k(\gamma^*)^{-2} + \tilde{C}'\right).$$

■

Theorem 3.4. *Suppose $\frac{B}{T} \geq \delta_{drift}$. Consider a* BwK *instance such that (i) for each arm and resource, the expected consumption of that resource differs from $\frac{B}{T}$ by at least δ_{drift}; and (ii) all the other assumptions required by Theorem 3.3 (Section 3.2.3) are also satisfied. Then, there is an algorithm for* BwK *whose regret satisfies the same bound as in Theorem 3.3 with the same constant $\tilde{C}$.*

Note that BwK is not automatically a special case of our model because of our assumption that the null arm has strictly positive drift for every resource. Now we present a reduction from BwK with $\frac{B}{T}$ bounded away from 0 to our model. We show that our results imply a logarithmic regret bound for BwK under certain assumptions.

Proof. **Reduction.** Assume we are given an instance of BwK with $\frac{B}{T} \geq \delta_{\text{drift}} > 0$. (Existing results on logarithmic regret for BwK also assume the ratio of the initial budget to the time horizon is bounded away from 0 [Li et al., 2021a].) We will reduce the given BwK instance to a problem in our model. The reduction initializes an instance of ExploreThenControlBudget (Algorithm 3) running in a simulated environment with the same set of arms as in the given BwK instance, plus an additional null arm whose drift is equal to δ_{drift} deterministically for each resource. The reduction will maintain two time counters: t_a is the actual number of time steps that have elapsed in the BwK problem, and t_s is the number of time steps that have elapsed in the simulation environment in which Algorithm 3 is running. Likewise, there are two vectors that track the remaining budget: B_a is the remaining budget in the actual BwK problem our reduction is solving, while B_s is the remaining budget in the simulation environment. These two budget vectors will always be related by the equation

$$B_s = B_a - T\delta_{\text{drift}}\mathbf{1} + t_s\delta_{\text{drift}}\mathbf{1}. \tag{3.21}$$

In particular, the initial budget of each resource is initialized (at simulated time $t_s = 0$) to $B - T\delta_{\text{drift}}$.

Each step of the reduction works as follows. We call Algorithm 3 to simulate one time step in the simulated environment. If Algorithm 3 recommends to pull a non-null arm x, we pull arm x, increment both of the time counters (t_a and t_s), and update the vector of remaining resource amounts, B_a, according to the resources consumed by arm x. If Algorithm 3 recommends to pull the null arm, we do not pull any arm, and we leave t_a and B_a unchanged; however, we still increment the simulated time counter t_s. Finally, regardless of whether a null or non-null arm was pulled, we update B_s to satisfy Eq. (3.21).

Correctness. Since the reduction pulls the same sequence of non-null arms as Algorithm 3 until the BwK stopping condition is met and the additional pulls of the null arm in the simulation environment yield zero reward, the total reward in the actual BwK problem equals the total reward earned in the simulation environment at the time when the BwK stopping condition is met and the reduction ceases running. Since Algorithm 3 maintains the invariant that B_s is a nonnegative vector, Eq. (3.21) ensures that B_a will also remain nonnegative as long as $t_s \geq T$ must hold. Theorem 3.3 ensures that the total expected reward earned in the simulation environment and hence, also in the BwK problem itself, is bounded below by $T \cdot \mathsf{OPT}_{\mathsf{LP}} - \tilde{C} \cdot \log T$, where $\tilde{C}$ is the constant in Theorem 3.3 and $\mathsf{OPT}_{\mathsf{LP}}$ denotes the optimal value of the LP relaxation (Eq. (3.1)) for the simulation environment.

We would like to show that this implies the regret of the reduction (with respect to the LP relaxation of BwK) is bounded by $\tilde{C} \cdot \log T$. To do so, we must show that the LP relaxations of the original BwK problem and the simulation environment have the same optimal value. Let μ_x^r and $\mu_x^{d,j}$ denote the expected reward and expected drifts in the actual BwK problem with arm set $\mathcal{X}$, and let $\hat{\mu}_x^r$ and $\hat{\mu}_x^{d,j}$ denote the expected reward and drifts in the simulation environment with arm set $\mathcal{X}^+ = \mathcal{X} \cup \{x^0\}$. The two LP formulations are as follows.

$$\begin{array}{llll} \max\limits_{p} & \sum\limits_{x \in \mathcal{X}} p_x \mu_x^r & & \\ \text{s.t.} & \sum\limits_{x \in \mathcal{X}} p_x \mu_x^{d,j} & \geq -\frac{B}{T} & \forall j \in \mathcal{J}, \\ & \sum\limits_{x \in \mathcal{X}} p_x & \leq 1 & \\ & p_x & \geq 0 & \forall x \in \mathcal{X}. \end{array} \qquad \begin{array}{llll} \max\limits_{p} & \sum\limits_{x \in \mathcal{X}} p_x \hat{\mu}_x^r & & \\ \text{s.t.} & \sum\limits_{x \in \mathcal{X}} p_x \hat{\mu}_x^{d,j} & \geq -\frac{B}{T} - \delta_{\text{drift}} & \forall j \in \mathcal{J}, \\ & \sum\limits_{x \in \mathcal{X}^+} p_x & = 1 & \\ & p_x & \geq 0 & \forall x \in \mathcal{X}^+. \end{array}$$

The differences between the two LP formulations lie in substituting $\hat{\mu}$ for μ, substituting $\mathcal{X}^+$ for $\mathcal{X}$, and transforming the inequality constraint $\sum_{x \in \mathcal{X}} p_x \leq 1$ into an equality constraint $\sum_{x \in \mathcal{X}^+} p_x = 1$. We know that $\mu_x^r = \hat{\mu}_x^r$ for every $x \in \mathcal{X}$ and $\hat{\mu}_{x^0}^r = 0$. Furthermore, $\hat{\mu}_x^{d,j}$ denotes the expected drift of resource j in the simulation environment when arm x is pulled. This can be written as the sum of two terms: drift $\mu_x^{d,j}$ is the expectation of the (non-positive) quantity added to the jth component of budget vector B_a when pulling arm x in the actual BwK environment; in addition to this non-positive drift, there is a deterministic positive drift of δ_{drift} due to incrementing the simulation time counter t_s and recomputing B_s using Eq. (3.21). Hence, $\hat{\mu}_x^{d,j} = \mu_x^{d,j} + \delta_{\text{drift}}$ for all $x \in \mathcal{X}$ and $j \in \mathcal{J}$. Furthermore, $\hat{\mu}_{x^0}^{d,j} = \delta_{\text{drift}}$. Hence, for any vector $\vec{p}$ representing a probability distribution on $\mathcal{X}^+$, we have

$$\sum_{x \in \mathcal{X}^+} p_x \hat{\mu}_x^{d,j} = \left(\sum_{x \in \mathcal{X}} p_x \hat{\mu}_x^{d,j} \right) + \delta_{\text{drift}}. \tag{3.22}$$

Accordingly, a vector $\vec{p}$ satisfies the constraints of the BwK LP relaxation above if and only if the probability vector on $\mathcal{X}^+$ obtained from $\vec{p}$ by setting $p_{x^0} = 1 - \sum_{x \in \mathcal{X}} p_x$ satisfies the constraints of the second LP relaxation above. This defines a one-to-one correspondence between the sets of vectors feasible for the two LP formulations. Furthermore, this one-to-one correspondence preserves the value of the objective function because $\hat{\mu}^r_x = \mu^r_x$ for $x \in \mathcal{X}$ and $\hat{\mu}^r_{x^0} = 0$. Thus, the optimal value of the two linear programs is the same. This completes the proof. ■

3.10 Detailed Assumptions about the Null Arm

In any model in which resources can be consumed and/or replenished over time, one must specify what happens when the budget of one (or more) resources reaches zero. The original bandits with knapsacks problem assumes than when this happens, the process of learning and gaining rewards ceases. The key distinction between that model and ours is that we instead assume the learner is allowed to remain idle until the supply of every resource becomes positive again, at which point the learning process recommences. The null arm in our chapter is intended to represent this option to remain idle and wait for resource replenishment. In order for these idle periods to have finite length almost surely, a minimal assumption is that when the null arm is pulled, for each resource there is a positive probability that the supply of the resource increases. We make the stronger assumption that for each resource, the expected change in supply is positive when the null arm is pulled. In fact, our results for the MDP setting hold under the following more general assumption: there exists a probability distribution over arms, such that when a random arm is sampled from this distribution and pulled, the expected change in the supply of each resource is positive. In the following, we refer to this as Assumption PD (for ”positive drift”).

To see that our results for the MDP setting continue to hold under Assumption PD (i.e., even if one doesn't assume that the null arm itself is guaranteed to yield positive expected drift for each resource) simply modify Algorithms 2 and 3 so that whenever they pull the null arm in a time step when the supply of each resource is at least 1, the modified algorithms instead pull a random arm sampled from the probability distribution over arms that guarantees positive expected drift for every resource. As long as the constant δ_{drift} is less than or equal to this positive expected drift, the modification to the algorithms does not change their analysis. We believe it's likely that our learning algorithm (Algorithm 3) could similarly be adapted to work under Assumption PD, but it would be less straightforward because the positive-drift distribution over arms would need to be learned.

When Assumption PD is violated, the problem becomes much more similar to the Bandits with Knapsacks problem. To see why, consider a two-player zero-sum game in which the row player chooses an arm x, the column player chooses a resource j, and the payoff is the expected drift of that resource when that arm is pulled, $\mu_x^{d,j}$. Assumption PD is equivalent to the assertion that the value of the game is positive; the negation of Assumption PD means that the value of the game is non-positive. By the Minimax Theorem, this means there is a convex combination of resources (i.e., a mixed strategy for the column player) such that the weighted-average supply of these resources is guaranteed to experience non-positive expected drift, no matter which arm is pulled. Either the expected drift is zero — we prove in the next section (Section 3.10) that regret $O(\sqrt{T})$ is unavoidable in this case — or the expected drift is strictly negative, in which case the weighted-average resource supply inevitably dwindles to zero no matter which arms the learner pulls. In either case, the behavior of the model is qualitatively different when Assumption PD does not hold.

3.11 Regret Bounds for One Arm, One Resource, and Zero Drift

In this section we will consider the case when $\mathcal{X} = \{x^0, x\}$, $\mathcal{J} = \{1\}$, and x has zero drift, i.e., $\mu_x^d = 0$. Since x is the only arm besides the null arm, we assume without loss of generality that its reward is equal to 1 deterministically. The optimal policy is to pull x^0 when $B_{t-1} < 1$ and x otherwise. We will show that the regret of this policy is $\Theta(\sqrt{T})$. Therefore, our separation assumption that $\delta_{\text{drift}} = \min\{|\mu_x^{d,j}| : x \in \mathcal{X}, j \in \mathcal{J}\} > 0$ is necessary for logarithmic regret bounds.

Theorem 3.5. *The regret of the MDP policy is $O(\sqrt{T})$.*

Proof. The optimal solution of the LP relaxation (Eq. (3.1)) is $p_x = 1$ and $p_{x^0} = 0$. Since x^0 and x have reward equal to 0 and 1 deterministically, $\mathsf{OPT}_{\mathsf{LP}} = 1$.Therefore, the regret of the MDP policy is equal to the expected number rounds in which the budget is less than 1. That is,

$$\mathsf{Reg}_T^{\mathsf{BwK}} = \mathbb{E}\left[\sum_{t=1}^{T} \mathbb{1}[B_{t-1} < 1]\right].$$

Since $\mathbb{E}[d_t] = 0$ when $B_{t-1} \geq 1$ and $\mathbb{E}[d_t] = \mu^d_{x^0}$ when $B_{t-1} < 1$, we can write

$$\begin{aligned}
\mathsf{Reg}^{\mathsf{BwK}}_T &= \mathbb{E}\left[\sum_{t=1}^{T} \mathbb{1}[B_{t-1} < 1]\right] \\
&= \frac{1}{\mu^d_{x^0}} \mu^d_{x^0} \mathbb{E}\left[\sum_{t=1}^{T} \mathbb{1}[B_{t-1} < 1]\right] \\
&= \frac{1}{\mu^d_{x^0}} \mathbb{E}\left[\sum_{t=1}^{T} \mu^d_{x^0} \mathbb{1}[B_{t-1} < 1] + 0\mathbb{1}[B_{t-1} \geq 1]\right] \\
&= \frac{1}{\mu^d_{x^0}} \mathbb{E}\left[\sum_{t=1}^{T} d_t\right] \\
&= \frac{1}{\mu^d_{x^0}} \left(\mathbb{E}\left[B_T\right] - B\right).
\end{aligned}$$

Since $B_0 = B$, the budget is updated as $B_t = B_{t-1} + d_t$ and $d_t \in [-1, 1]$, we have

$$\begin{aligned}
\mathbb{E}\left[B_t^2 | B_{t-1}\right] &= \mathbb{E}\left[B_{t-1}^2 + 2B_{t-1}d_t + d_t^2 | B_{t-1}\right] \\
&= \mathbb{E}\left[B_{t-1}^2 | B_{t-1}\right] + \mathbb{E}\left[2B_{t-1}d_t | B_{t-1}\right] + \mathbb{E}\left[d_t^2 | B_{t-1}\right] \\
&\leq B_{t-1}^2 + \mathbb{E}\left[2B_{t-1}d_t | B_{t-1}\right] + 1^2 \\
&= B_{t-1}^2 + 2B_{t-1}\mu^d_{x^0} \mathbb{1}[B_{t-1} < 1] + 1 \\
&\leq B_{t-1}^2 + 2\mu^d_{x^0} + 1 \\
\Rightarrow \mathbb{E}\left[B_T^2\right] &= O(T).
\end{aligned}$$

Using Jensen's inequality, we have

$$\mathbb{E}\left[B_T\right] \leq \sqrt{\mathbb{E}\left[B_T^2\right]} = O(\sqrt{T}).$$

This completes the proof. ■

Theorem 3.6. *If* $\mathbb{E}\left[d_t^2 | B_{t-1}\right] \geq \sigma^2 > 0$, *then the regret of the MDP policy is* $\Omega(\sqrt{T})$.

Proof. Using the proof of Theorem 3.5, it suffices to provide a lower bound on $\mathbb{E}[B_T]$. Since the budget is updated as $B_t = B_{t-1} + d_t$, $\mathbb{E}[d_t|B_{t-1}] \geq 0$, and $\mathbb{E}\left[d_t^2|B_{t-1}\right] \geq \sigma^2$, we have

$$\begin{aligned}
\mathbb{E}\left[B_t^2|B_{t-1}\right] &= \mathbb{E}\left[B_{t-1}^2 + 2B_{t-1}d_t + d_t^2|B_{t-1}\right] \\
&= \mathbb{E}\left[B_{t-1}^2|B_{t-1}\right] + \mathbb{E}[2B_{t-1}d_t|B_{t-1}] + \mathbb{E}\left[d_t^2|B_{t-1}\right] \\
&\geq B_{t-1}^2 + \mathbb{E}[2B_{t-1}d_t|B_{t-1}] + \sigma^2 \\
&= B_{t-1}^2 + 2B_{t-1}\mathbb{E}[d_t|B_{t-1}] + \sigma^2 \\
&\geq B_{t-1}^2 + \sigma^2 \\
\Rightarrow \mathbb{E}\left[B_T^2\right] &\geq \Omega(T).
\end{aligned}$$

The Cauchy-Schwarz inequality yields that

$$\mathbb{E}\left[\left(B_T^{1/2}\right)^2\right]^{1/2} \mathbb{E}\left[\left(B_T^{3/2}\right)^2\right]^{1/2} \geq \mathbb{E}\left[B_T^2\right] \geq \Omega(T).$$

Squaring both sides yields that

$$\mathbb{E}[B_T]\,\mathbb{E}\left[B_T^3\right] \geq \mathbb{E}\left[B_T^2\right]^2 \geq \Omega(T^2).$$

It suffices to show that $\mathbb{E}\left[B_T^3\right] = O(T^{3/2})$ because this will imply that $\mathbb{E}[B_T] = \Omega(T^{1/2})$. Since $d_t \in [-1, 1]$, we have

$$\begin{aligned}
&\mathbb{E}\left[B_t^3|B_{t-1}\right] \\
&= \mathbb{E}\left[B_{t-1}^3 + 3B_{t-1}^2 d_t + 3B_{t-1}d_t^2 + d_t^3|B_{t-1}\right] \\
&= B_{t-1}^3 + 3B_{t-1}^2\mathbb{E}[d_t|B_{t-1}] + 3B_{t-1}\mathbb{E}\left[d_t^2|B_{t-1}\right] + \mathbb{E}\left[d_t^3|B_{t-1}\right] \\
&= B_{t-1}^3 + 3B_{t-1}^2\mu_{x^0}^d \mathbb{1}[B_{t-1} < 1] + 3B_{t-1}\mathbb{E}\left[d_t^2|B_{t-1}\right] + \mathbb{E}\left[d_t^3|B_{t-1}\right] \\
&\leq B_{t-1}^3 + 3\mu_{x^0}^d + 3B_{t-1} + 1.
\end{aligned}$$

Taking expectation on both sides yields

$$\begin{aligned}\mathbb{E}\left[B_t^3\right] &\leq \mathbb{E}\left[B_{t-1}^3\right] + 3\mathbb{E}\left[B_{t-1}\right] + O(1)\\ &\leq \mathbb{E}\left[B_{t-1}^3\right] + O(\sqrt{t}),\end{aligned}$$

where the last inequality follows from the proof of Theorem 3.5 where we show that $\mathbb{E}\left[B_t\right] \leq O(\sqrt{t})$. Summing both sides over all rounds yields that

$$\mathbb{E}\left[B_T^3\right] = O(T^{3/2})$$

and this completes the proof. ■

CHAPTER 4

ONLINE CONVEX OPTIMIZATION WITH UNBOUNDED MEMORY

In this chapter we focus on a different model of online decision-making, namely, online convex optimization. Online convex optimization (OCO) is a widely used framework in online learning. In each round, the learner chooses a decision in a convex set and an adversary chooses a convex loss function, and then the learner suffers the loss associated with their current decision. However, in many applications the learner's loss depends not only on the current decision but on the entire history of decisions until that point. The OCO framework and its existing generalizations do not capture this, and they can only be applied to many settings of interest after a long series of approximation arguments. They also leave open the question of whether the dependence on memory is tight because there are no non-trivial lower bounds. In this chapter we introduce a generalization of the OCO framework, "Online Convex Optimization with Unbounded Memory", that captures long-term dependence on past decisions. We introduce the notion of p-effective memory capacity, H_p, that quantifies the maximum influence of past decisions on present losses. We prove an $O(\sqrt{H_p T})$ upper bound on the policy regret and a matching (worst-case) lower bound. As a special case, we prove the first non-trivial lower bound for OCO with finite memory [Anava et al., 2015], which could be of independent interest, and also improve existing upper bounds. We demonstrate the broad applicability of our framework by using it to derive regret bounds, and to improve and simplify existing regret bound derivations, for a variety of online learning problems including online linear control and an online variant of performative prediction. This chapter is based on joint work with Sarah Dean and Robert Kleinberg [Kumar et al., 2023].

4.1 Introduction

Numerous applications are characterized by multiple rounds of sequential interactions with an environment, e.g., prediction from expert advice [Littlestone and Warmuth, 1989, 1994], portoflio selection [Cover, 1991], routing [Awerbuch and Kleinberg, 2008], etc. One of the most popular frameworks for modelling such sequential decision-making problems is online convex optimization (OCO) [Zinkevich, 2003]. The OCO framework is as follows. In each round, the learner chooses a decision in a convex set and an adversary chooses a convex loss function, and then the learner suffers the loss associated with their current decision. The performance of an algorithm is measured by regret: the difference between the algorithm's total loss and that of the best fixed decision. We refer the reader to Shalev-Shwartz [2012], Hazan [2022], Orabona [2019] for surveys on this topic.

However, in many applications the loss of the learner depends not only on the current decisions but on the entire history of decisions until that point. For example, in online linear control [Agarwal et al., 2019b], in each round the learner chooses a "control policy" (i.e., decision), suffers a loss that is a function of the action taken by this policy and the current state of the system, and the system's state evolves according to linear dynamics. The current state depends on the entire history of actions and, therefore, the current loss depends not only on the current decision but the entire history of decisions. The OCO framework cannot capture such long-term dependence of the current loss on the past decisions and neither can existing generalizations that allow the loss to depend on a *constant* number of past decisions [Anava et al., 2015]. Although a series of approximation arguments can be used to apply finite memory generalizations of OCO to the online linear control problem, there is no OCO framework that captures the complete long-term dependence of current losses on past decisions. Furthermore, there are no non-trivial lower bounds for OCO in the memory setting,[1] which leaves open the question whether the dependence on memory is tight.

Contributions. In this chapter we introduce a generalization of the OCO framework, "Online Convex Optimization with Unbounded Memory" (Section 4.2), that allows the loss in the current round to depend on the entire history of decisions until that point. We introduce the notion of p-effective memory capacity, H_p, that quantifies the maximum influence of past decisions on present losses. We prove an $O(\sqrt{H_p T})$ upper bound on the policy regret (Theorem 4.2) and a matching (worst-case) lower bound (Theorem 4.3). As a special case, we prove the first non-trivial lower bound for OCO with finite memory (Theorem 4.5), which could be of independent interest, and also improve existing upper bounds (Theorem 4.4). Our lower bound technique extends existing techniques developed for memoryless settings. We design novel adversarial loss functions that exploit the

[1]The trivial lower bound refers to the $\Omega(\sqrt{T})$ lower bound for OCO in the memoryless setting.

fact that an algorithm cannot overwrite its history. We illustrate the power of our framework by bringing together the regret analysis of two seemingly disparate problems under the same umbrella. First, we show how our framework improves and simplifies existing regret bounds for the online linear control problem [Agarwal et al., 2019b] in Theorem 4.8. Second, we show how our framework can be used to derive regret bounds for an online variant of performative prediction [Perdomo et al., 2020] in Theorem 4.9. This demonstrates the broad applicability of our framework for deriving regret bounds for a variety of online learning problems, particularly those that exhibit long-term dependence of current losses on past decisions.

Related work. The most closely related work to ours is the OCO with finite memory framework [Anava et al., 2015]. They consider a generalization of the OCO framework that allows the current loss to depend on a *constant* number of past decisions. There have been a number of follow-up works that extend the framework in a variety of other ways, such as non-stationarity [Zhao et al., 2022], incorporating switching costs [Shi et al., 2020], etc. However, none of these existing works go beyond a constant memory length and do not prove a non-trivial lower bound with a dependence on the memory length. In a different line of work, Bhatia and Sridharan [2020] consider a much more general online learning framework that goes beyond a constant memory length, but they only provide *non-constructive* upper bounds on regret. In contrast, our OCO with unbounded memory framework allows the current loss to depend on an *unbounded* number of past decisions, provides *constructive* upper bounds on regret, and lower bounds for a broad class of problems that includes OCO with finite memory with a general memory length m.

A different framework for sequential decision-making is multi-armed bandits [Bubeck and Cesa-Bianchi, 2012, Slivkins, 2019]. Qin et al. [2023] study a variant of contextual stochastic bandits where the current loss can depend on a sparse subset of all prior contexts. This setting differs from ours due to the feedback model, stochasticity, and decision space. Reinforcement learning [Sutton and Barto, 2018] is yet another popular framework for sequential decision-making that considers very general state-action models of feedback and dynamics. In reinforcement learning one typically measures regret with respect to the best state-action policy from some policy class, rather than the best fixed decision as in online learning and OCO. In the special case of linear control, policies can be reformulated as decisions while preserving convexity; we discuss this application in Section 4.4. Considering the general framework is an active area of research.

We defer discussion of related work for specific applications to Section 4.4.

4.2 Framework

We begin with some motivation for the formalism used in our framework (Section 4.2.1). Many real-world applications involve controlling a physical dynamical system, for example, variable-speed wind turbines in wind energy electric power production [Boukhezzar and Siguerdidjane, 2010]. The typical solution for these problems has been to model them as offline control problems with linear time-invariant dynamics and use classical methods such as LQR and LQG [Boukhezzar and Siguerdidjane, 2010]. Instead of optimizing over the space of control inputs, the typical feedback control approach optimizes over the space of controllers, i.e., policies that choose a control input as a function of the system state. The standard controllers considered in the literature are linear controllers. Even when the losses are convex in the state and input, they are nonconvex in the linear controller. In the special case of quadratic losses in terms of the state and input, there is a closed-form solution for the optimal solution using the algebraic Riccati equations [Lancaster and Rodman, 1995]. But this does not hold for general convex losses resulting in convex reparameterizations such as Youla [Youla et al., 1976, Kučera, 1975] and SLS [Wang et al., 2019, Anderson et al., 2019]. The resulting parameterization represents an infinite dimensional system response and is characterized by a sequence of matrices. Recent work has studied an online approach for some of these control theory problems, where a sequence of controllers is chosen adaptively rather than choosing one offline [Abbasi-Yadkori and Szepesvári, 2011, Dean et al., 2018, Simchowitz and Foster, 2020, Agarwal et al., 2019b].

The takeaway from the above is that there are online learning problems in which (i) the current loss depends on the entire history of decisions; and (ii) the decision space can be more complicated than just a subset of $\mathbb{R}^d$, e.g., it can be an unbounded sequence of matrices. This motivates us to model the decision space as a Hilbert space and the history space as a Banach space in the formal problem setup below, and this subsumes the special cases of OCO and OCO with finite memory. This formalism not only lets us consider a wide range of spaces, such as $\mathbb{R}^d$, unbounded sequences of matrices, etc., but also lets us define appropriate norms on these spaces. This latter feature is crucial for deriving strong regret bounds for some applications such as online linear control. For this problem we derive improved regret bounds (Theorem 4.8) by defining weighted norms on the decision and history spaces, where the weights are chosen to leverage the problem structure.

Notation. We use $\|\cdot\|_{\mathcal{U}}$ to denote the norm associated with a space $\mathcal{U}$. The operator norm for a linear operator L from space $\mathcal{U} \to \mathcal{V}$ is defined as $\|L\|_{\mathcal{U}\to\mathcal{V}} = \max_{u:\|u\|_{\mathcal{U}}\leq 1} \|Lu\|_{\mathcal{V}}$. For convenience, sometimes we simply use $\|\cdot\|$ when the meaning is clear from the context. For a finite-dimensional matrix we use $\|\cdot\|_F$ and $\|\cdot\|_2$ to denote its Frobenius norm and operator norm respectively. In this chapter we use $O(\cdot)$ to hide absolute constants only; we clarify inline when we hide other terms as in Section 4.5.

4.2.1 Setup

Let the decision space $\mathcal{X}$ be a closed and convex subset of a Hilbert space $\mathcal{W}$ with norm $\|\cdot\|_{\mathcal{X}}$ and the history space $\mathcal{H}$ be a Banach space with norm $\|\cdot\|_{\mathcal{H}}$. Let

$A : \mathcal{H} \to \mathcal{H}$ and $B : \mathcal{W} \to \mathcal{H}$ be linear operators. The game between the learner and an oblivious adversary proceeds as follows. Let T denote the time horizon and $f_t : \mathcal{H} \to \mathbb{R}$ be loss functions chosen by the adversary. The initial history is $h_0 = 0$. In each round $t \in [T]$, the learner chooses $x_t \in \mathcal{X}$, the history is updated to $h_t = Ah_{t-1} + Bx_t$, and the learner suffers loss $f_t(h_t)$. An instance of an online convex optimization with unbounded memory problem is specified by the tuple $(\mathcal{X}, \mathcal{H}, A, B)$.

We use the notion of policy regret [Dekel et al., 2012] as the performance measure in our framework. The policy regret of a learner is the difference between its total loss and the total loss of a strategy that plays the best fixed decision in every round. The history after round t for a strategy that chooses x in every round is described by $h_t = \sum_{k=0}^{t-1} A^k Bx$, which motivates the following definition.

Definition 4.2.1. Given $f_t : \mathcal{H} \to \mathbb{R}$, the function $\tilde{f}_t : \mathcal{X} \to \mathbb{R}$ is defined as

$$\tilde{f}_t(x) = f_t(\sum_{k=0}^{t-1} A^k Bx).$$

Definition 4.2.2 (Policy Regret)**.** The policy regret of an algorithm $\mathcal{A}$ is defined as

$$\mathsf{Reg}_T^{\mathsf{OCO-UM}}(\mathcal{A}) = \sum_{t=1}^{T} f_t(h_t) - \min_{x \in \mathcal{X}} \sum_{t=1}^{T} \tilde{f}_t(x).$$

In many motivating examples such as online linear control (Section 4.4.1), the history at the end of a round is a sequence of linear transformations of past decisions. The following definition captures this formally and we leverage this structure to prove stronger regret bounds (Theorem 4.2).

Definition 4.2.3 (Linear Sequence Dynamics)**.** Consider an online convex optimization with unbounded memory problem specified by $(\mathcal{X}, \mathcal{H}, A, B)$. Let $(\xi_k)_{k=0}^{\infty}$ be a sequence of nonnegative real numbers satisfying $\xi_0 = 1$. We say that $(\mathcal{X}, \mathcal{H}, A, B)$ follows linear sequence dynamics with the ξ-weighted p-norm for $p \geq 1$ if

1. $\mathcal{H}$ is the ξ-weighted ℓ^p-direct sum of a finite or countably infinite number of copies of $\mathcal{W}$: every element $y \in \mathcal{H}$ is a sequence $y = (y_i)_{i \in \mathcal{I}}$, where $\mathcal{I} = \mathbb{N}$ or $\mathcal{I} = \{0, \dots, n\}$ for some $n \in N$, and $\|y\|_{\mathcal{H}} = \left(\sum_{i \in \mathcal{I}} (\xi_i \|y_i\|)^p\right)^{1/p} < \infty$.
2. We have $A(y_0, y_1, \dots) = (0, A_0 y_0, A_1 y_1, \dots)$, where $A_i : \mathcal{W} \to \mathcal{W}$ are linear operators.
3. The operator B satisfies $B(x) = (x, 0, \dots)$.

Note that since the norm on $\mathcal{H}$ depends on the weights ξ, the operator norm $\|A^k\|$ also depends on ξ. If the weights are all equal to 1, then we simply say p-norm instead of ξ-weighted p-norm.

Formulation as an online decision-making problem. Before continuing, we take a brief detour to show that the problem defined above can be formulated in the online decision-making framework (Definition 2.1.6), which is defined by the tuple $(\mathcal{X}, \Psi, \mathcal{L}, \mathcal{G}, \Phi, \phi, \Pi, \mathsf{ENV})$.

- Actions $\mathcal{X}$ are the same as above.
- States $\Psi = \mathcal{H}$.

- Initial state $\psi_0 = h_0$.
- Dynamics function $g_t = g$ for all rounds, where

$$g(\psi_{t-1}, x_t) = g(h_{t-1}, x_t) = Ah_{t-1} + Bx_t.$$

  Here, $\mathcal{G}$ consists of a single function defined by the previous equation.
- Loss function $l_t = f_t$ for all rounds. Here, $\mathcal{L}$ is the set of functions from $\mathcal{H} \to \mathbb{R}$.
- Feedback spaces $\Phi_l = \mathcal{L}$ and $\Phi_g = \mathcal{G}$.
- Feedback functions $\phi_l(l, \psi_t) = l$ and $\phi_g(g, \psi_{t-1}, x_t) = g$.
- Decisions Π are constant functions taking values in the set of point-mass distributions over $\mathcal{X}$.
- The environment ENV is an oblivious adversary.

We remark that decisions correspond to actions in the above formalism. However, our OCO with unbounded memory framework is rich enough to model applications where decisions are more general policies, such as online linear control (Section 4.4.1). For simplicity, in the rest of this chapter we will use the formalism and notation defined previously instead of the one above.

4.2.2 Assumptions

We make the following assumptions about the feedback model and the loss functions.

A1 The learner knows the operators A and B, and observes f_t at the end of each round t.

A2 The operator norm of B is at most 1, i.e., $\|B\| \leq 1$.

A3 The functions f_t are convex.

A4 The functions f_t are L-Lipschitz continuous: $\forall\ h, \tilde{h} \in \mathcal{H}$ and $t \in [T]$, we have $|f_t(h) - f_t(\tilde{h})| \leq L\|h - \tilde{h}\|_{\mathcal{H}}$.

Regarding Assumption **A1**, our results easily extend to the case where instead of observing f_t, the learner receives a gradient $\nabla \tilde{f}_t(x_t)$ from a gradient oracle, which can be implemented using knowledge of f_t and the dynamics A and B. Handling the cases when the operators A and B are unknown and/or the learner observes bandit feedback (i.e., only $f_t(h_t)$) are important problems and we leave them as future work. Note that our assumption that A and B are known is no more restrictive than in the existing literature on OCO with finite memory [Anava et al., 2015] where it is assumed that the learner knows the constant memory length. In fact, our assumption is more general because our framework not only captures constant memory length as a special case but allows for richer dynamics as we illustrate in Section 4.4. Assumption **A2** is made for convenience, and it amounts to a rescaling of the problem. Assumption **A3** can be replaced by the *weaker* assumption that $\tilde{f}_t$ are convex (similar to the literature on OCO with finite memory [Anava et al., 2015]) and this is what we use in the rest of the chapter.

Assumptions **A1** and **A4** imply that $\tilde{f}_t$ are $\tilde{L}$-Lipschitz continuous for the following $\tilde{L}$.

Theorem 4.1. *Consider an online convex optimization with unbounded memory problem specified by* $(\mathcal{X}, \mathcal{H}, A, B)$*. If* f_t *is L-Lipschitz continuous,*

$$\tilde{L} \leq L \sum_{k=0}^{\infty} \|A^k\|.$$

If $(\mathcal{X}, \mathcal{H}, A, B)$ *follows linear sequence dynamics with the* ξ*-weighted p-norm for* $p \geq 1$*, then*

$$\tilde{L} \leq L \left(\sum_{k=0}^{\infty} \|A^k\|^p \right)^{\frac{1}{p}}.$$

The proof follows from the definitions of $\tilde{f}_t$ and $\| \cdot \|_{\mathcal{H}}$, and we defer it to Section 4.9. The above bound is tighter than similar results in the literature on OCO with finite memory and online linear control. This theorem is a key ingredient, amongst others, in improving existing upper bounds on regret for OCO with finite memory (Theorem 4.4) and for online linear control (Theorem 4.8). Before presenting our final assumption we introduce the notion of p-effective memory capacity that quantifies the maximum influence of past decisions on present losses.

Definition 4.2.4 (p-Effective Memory Capacity)**.** Consider an online convex optimization with unbounded memory problem specified by $(\mathcal{X}, \mathcal{H}, A, B)$. For $p \geq 1$, the p-effective memory capacity is defined as

$$H_p(\mathcal{X}, \mathcal{H}, A, B) = \left(\sum_{k=0}^{\infty} k^p \|A^k\|^p \right)^{\frac{1}{p}}. \tag{4.1}$$

When the meaning is clear from the context we simply use H_p instead. The p-effective memory capacity is an upper bound on the difference in histories for two sequences of decisions whose difference grows at most linearly with time. To see this, consider two sequences of decisions, (x_k) and $(\tilde{x}_k)$, whose elements differ by no more than k at time k: $\|x_k - \tilde{x}_k\| \leq k$. Then the histories generated by the two sequences have difference between bounded as $\|h - \tilde{h}\| = \| \sum_k A^k B(x_k - \tilde{x}_k)\| \leq \sum_k k\|A^k B\| \leq \sum_k k\|A^k\| = H_1$, where the last inequality follows from Assumption **A2**. A similar bound holds with H_p instead when $(\mathcal{X}, \mathcal{H}, A, B)$ follows linear sequence dynamics with the ξ-weighted p-norm.

A5 The 1-effective memory capacity is finite, i.e., $H_1 < \infty$.

Since H_p is decreasing in p, $H_1 < \infty$ implies $H_p < \infty$ for all $p \geq 1$. For the case of linear sequence dynamics with the ξ-weighted p-norm it suffices to make the *weaker* assumption that $H_p < \infty$. However, for simplicity of exposition, we assume that $H_1 < \infty$.

4.2.3 Special Cases

OCO with Finite Memory. Consider the OCO with finite memory problem with constant memory length m. It can be specified in our framework by $(\mathcal{X}, \mathcal{H}, A_{\text{finite},m}, B_{\text{finite},m})$, where $\mathcal{H}$ is the ℓ^2-direct sum of m copies of $\mathcal{X}$, $A_{\text{finite},m}(x^{[m]}, \dots, x^{[1]}) = (0, x^{[m]}, \dots, x^{[2]})$, and $B_{\text{finite},m}(x) = (x, 0, \dots, 0)$. Note that $(\mathcal{X}, \mathcal{H}, A_{\text{finite},m}, B_{\text{finite},m})$ follows linear sequence dynamics with the 2-norm. Our framework can even model an extension where the problem follows linear sequence dynamics with the p-norm for $p \geq 1$ by simply defining $\mathcal{H}$ to be the ℓ^p-direct sum of m copies of $\mathcal{X}$.

OCO with ρ-discounted Infinite Memory. Our framework can also model OCO with infinite memory problems that are not modelled by existing OCO frameworks. Let $\rho \in (0,1)$ be the discount factor and $p \geq 1$. An OCO with ρ-discounted infinite memory problem is specified by $(\mathcal{X}, \mathcal{H}, A_{\text{infinite},\rho}, B_{\text{infinite},\rho})$, where $\mathcal{H}$ is the ℓ^p-direct sum of countably many copies of $\mathcal{X}$, $A_{\text{infinite},\rho}((y_0, y_1, \dots)) = (0, \rho y_0, \rho y_1, \dots)$, and $B_{\text{infinite},\rho}(x) = (x, 0, \dots)$. Note that $(\mathcal{X}, \mathcal{H}, A_{\text{infinite},\rho}, B_{\text{infinite},\rho})$ follows linear sequence dynamics with the p-norm.

4.3 Regret Analysis

We present two algorithms for choosing the decisions x_t. Algorithm 4 uses follow-the-regularized-leader (FTRL) [Shalev-Shwartz and Singer, 2006, Abernethy et al., 2008] on the loss functions $\tilde{f}_t$. We defer additional discussion of this algorithm to later in the chapter: we discuss how to implement it efficiently in Section 4.5 and present simple simulation experiments in Section 4.7. Algorithm 5, which we only present in Section 4.6, combines FTRL with a mini-batching approach [Dekel et al., 2012, Altschuler and Talwar, 2018, Chen et al., 2020] to additionally guarantee that the decisions switch at most $O(T\tilde{L}/LH_1)$ times. We defer the proofs of the following upper and lower bounds to Sections 4.10 and 4.11 respectively.

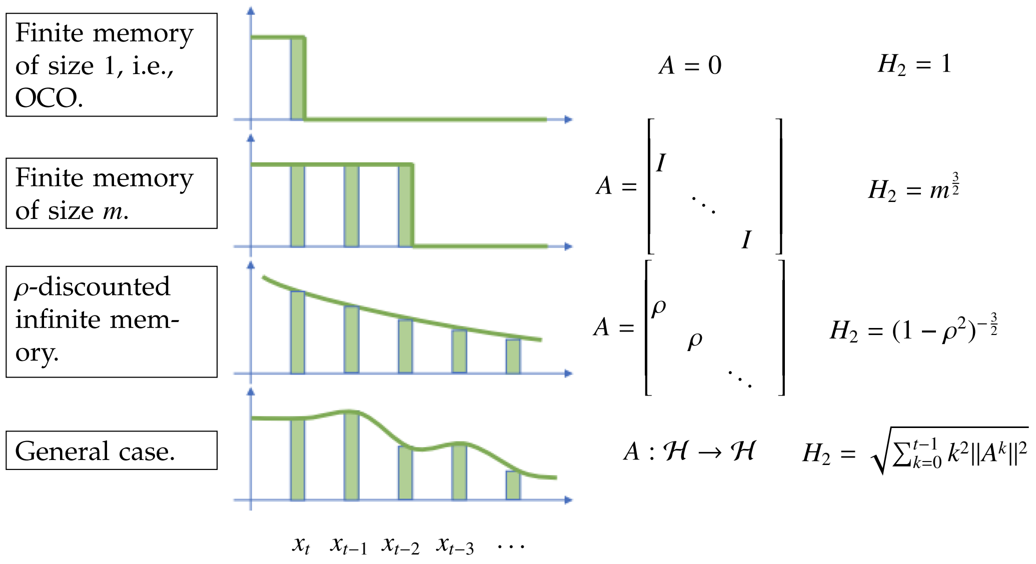

Finite memory of size 1, i.e., OCO.
Finite memory of size m.
ρ-discounted infinite memory.
General case.
x_t x_{t-1} x_{t-2} x_{t-3} $\cdots$
$A = 0$
$H_2 = 1$
$A = \begin{bmatrix} I & & \\ & \ddots & \\ & & I \end{bmatrix}$
$H_2 = m^{\frac{3}{2}}$
$A = \begin{bmatrix} \rho & & \\ & \rho & \\ & & \ddots \end{bmatrix}$
$H_2 = (1-\rho^2)^{-\frac{3}{2}}$
$A : \mathcal{H} \to \mathcal{H}$
$H_2 = \sqrt{\sum_{k=0}^{t-1} k^2 \|A^k\|^2}$

Figure 4.1: An illustration of what the dependence on past decisions looks like, the operator A, and the 2-effective memory capacity for some special cases and the general case. The bar chart is a cartoon illustration of the dependence on past decisions. For example, for OCO with finite memory of size m in the second row (for $m = 3$), the current loss depends on the past three decisions; for OCO with ρ-discounted infinite memory in the third row, the current loss depends on all past decisions with geometric decay of ρ; etc.

Theorem 4.2. *Consider an online convex optimization with unbounded memory problem specified by $(\mathcal{X}, \mathcal{H}, A, B)$. Let the regularizer $R : \mathcal{X} \to \mathbb{R}$ be α-strongly-convex and satisfy $|R(x) - R(\tilde{x})| \leq D$ for all $x, \tilde{x} \in \mathcal{X}$. Algorithm 4 with step-size η satisfies*

$$\mathsf{Reg}_T^{\mathsf{OCO-UM}}(\mathsf{FTRL}) \leq \frac{D}{\eta} + \eta\frac{T\tilde{L}^2}{\alpha} + \eta\frac{TL\tilde{L}H_1}{\alpha}.$$

If $\eta = \sqrt{\frac{\alpha D}{T\tilde{L}(LH_1+\tilde{L})}}$, then

$$\mathsf{Reg}_T^{\mathsf{OCO-UM}}(\mathsf{FTRL}) \leq O\left(\sqrt{\frac{D}{\alpha}TL\tilde{L}H_1}\right).$$

When $(\mathcal{X}, \mathcal{H}, A, B)$ follows linear sequence dynamics with the ξ-weighted p-norm, then all of the above hold with H_p instead of H_1.

Algorithm 4: FTRL

```
    Input  : Time horizon $T$, step size $\eta$, $\alpha$-strongly-convex regularizer
             $R : \mathcal{X} \to \mathbb{R}$.
  1 Initialize history $h_0 = 0$.
  2 for $t = 1, 2, \dots, T$ do
  3     Learner chooses $x_t \in \arg\min_{x \in \mathcal{X}} \sum_{s=1}^{t-1} \tilde{f}_s(x) + \frac{R(x)}{\eta}$.
  4     Set $h_t = Ah_{t-1} + Bx_t$.
  5     Learner suffers loss $f_t(h_t)$ and observes $f_t$.
  6 end
```

The proof of this theorem involves writing the regret as $\sum_t f_t(h_t) - \tilde{f}_t(x_t) + \sum_t \tilde{f}_t(x_t) - \tilde{f}_t(x^*)$, and bounding the first term using the p-effective memory capacity and the second term using (standard) FTRL for OCO without memory. We defer the full proof to Section 4.10. The following lower bound shows that this is tight in the worst-case.

Theorem 4.3. *There exists an instance of the online convex optimization with unbounded memory problem, $(\mathcal{X}, \mathcal{H}, A, B)$, that follows linear sequence dynamics with the ξ-weighted p-norm and there exist L-Lipschitz continuous loss functions $\{f_t : \mathcal{H} \to \mathbb{R}\}_{t=1}^T$ such that the regret of any algorithm $\mathcal{A}$ satisfies*

$$\mathsf{Reg}_T^{\mathsf{OCO-UM}}(\mathcal{A}) \geq \Omega\left(\sqrt{TL\tilde{L}H_p}\right).$$

The proof of this theorem follows from our lower bound for the special case of OCO with finite memory (Theorem 4.5), which we discuss in detail below. However, as we show in Section 4.11, the lower bound holds for a much broader class of problems.

Specialization to OCO with finite memory. Now we show how our bounds specialize to the special case of OCO with finite memory (Section 4.2.3).

Theorem 4.4. *Consider an online convex optimization with finite memory problem with constant memory length m specified by $(\mathcal{X}, \mathcal{H} = \mathcal{X}^m, A_{\text{finite},m}, B_{\text{finite},m})$. Let the regularizer $R : \mathcal{X} \to \mathbb{R}$ be α-strongly-convex and satisfy $|R(x) - R(\tilde{x})| \leq D$ for all $x, \tilde{x} \in \mathcal{X}$. Algorithm 4 with step-size $\eta = \sqrt{\frac{\alpha D}{T\tilde{L}(Lm^{\frac{3}{2}} + \tilde{L})}}$ satisfies*

$$\mathsf{Reg}_T^{\mathsf{OCO-UM}}(\mathsf{FTRL}) \leq O\left(\sqrt{\frac{D}{\alpha} TL\tilde{L}m^{\frac{3}{2}}}\right) \leq O\left(m\sqrt{\frac{D}{\alpha} TL^2}\right).$$

This follows from using the definition of $A_{\text{finite},m}$ to bound $\tilde{L}$ and H_2 in Theorem 4.2. This improves existing results [Anava et al., 2015] by a factor of $m^{1/4}$. Our bound depends on the Lipschitz continuity constants as $\sqrt{L\tilde{L}}$ whereas existing bounds depend as $\tilde{L}$, and $\tilde{L}$ can be as large as $\sqrt{m}L$ (Theorem 4.1). We defer the full proof to Section 4.10 and present a detailed comparison with existing results after the proof. (See this paragraph.)[2] The following lower bound shows that this is tight in the worst-case.

Theorem 4.5. *There exists an instance of the online convex optimization with finite memory problem with constant memory length m, $(\mathcal{X}, \mathcal{H} = \mathcal{X}^m, A_{\text{finite},m}, B_{\text{finite},m})$, and there exist L-Lipschitz continuous loss functions $\{f_t : \mathcal{H} \to \mathbb{R}\}_{t=1}^T$ such that the regret of any algorithm $\mathcal{A}$ satisfies*

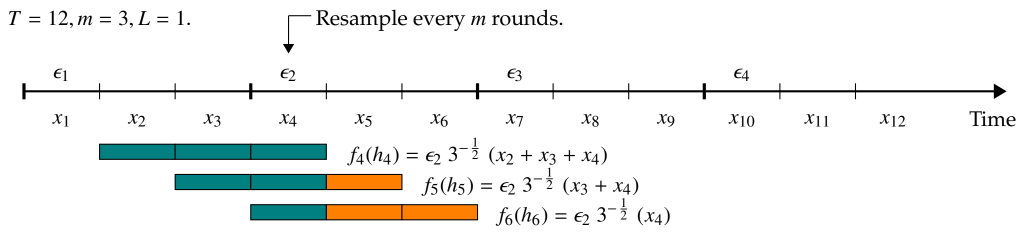
$$\mathsf{Reg}_T^{\mathsf{OCO-UM}}(\mathcal{A}) \geq \Omega\left(m\sqrt{TL^2}\right).$$

[2] A similar bound was independently obtained by Zhao et al. [2022, Theorem 20, Appendix B.8].

To the best of our knowledge, this is the first non-trivial lower bound for OCO with finite memory with an explicit dependence on the memory length m. Zhao et al. [2022] show a lower bound on an OCO with finite memory problem with *memory size 2*. They do so by constructing loss functions whose *Lipschitz constant depends on m*. Theorem 4.5 is the first lower bound for OCO with finite memory with *memory size m for general m* and *Lipschitz constant independent of m*. Our construction involves three main steps, the first two of which are loosely based on Altschuler and Talwar [2018]. First, divide time into $N = {}^{T}/_{m}$ blocks of size m. (For simplicity, assume T is a multiple of m.) Second, sample a random sign ϵ_n for each block $n \in [N]$. Third, for $t > m$ choose

$$f_t(h_t) = \underbrace{\epsilon_{\lceil \frac{t}{m} \rceil}}_{(a)} \underbrace{Lm^{-\frac{1}{2}}}_{(b)} \underbrace{\left(x_{t-m+1} + \cdots + x_{m\lfloor \frac{t}{m} \rfloor + 1}\right)}_{(c)},$$

where term (a) is the random sign ϵ_n sampled for the block $n = \lceil {}^{t}/_{m} \rceil$ that t belongs to, term (b) is a scaling factor chosen while respecting the Lipschitz continuity constraint, and term (c) is a sum over a *subset* of past decisions. Two important features of this construction are: (i) a random sign is sampled for each block rather than each round; and (ii) the loss in round t depends on the history of decisions until and including the first round of the block that t belongs to. These exploit the fact that an algorithm cannot overwrite its history and penalize it for its past decisions even after it observes the random sign ϵ_n for the current block. (See Fig. 4.2 for an illustration.) Existing lower bound proofs for OCO sample a random sign in each round and choose $f_t(x_t) \propto \epsilon_t x_t$. A first attempt at extending this for the OCO with finite memory setting would be to choose $f_t(h_t) \propto \epsilon_t \sum_{k=0}^{m-1} x_{t-k}$. However, in constrast to our approach, this does not exploit the fact that an algorithm cannot overwrite its history and does not suffice for obtaining a matching lower bound.



$T = 12, m = 3, L = 1.$
Resample every m rounds.
ϵ_1
ϵ_2
ϵ_3
ϵ_4
x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8 x_9 x_{10} x_{11} x_{12}
Time
$f_4(h_4) = \epsilon_2\, 3^{-\frac{1}{2}}\, (x_2 + x_3 + x_4)$
$f_5(h_5) = \epsilon_2\, 3^{-\frac{1}{2}}\, (x_3 + x_4)$
$f_6(h_6) = \epsilon_2\, 3^{-\frac{1}{2}}\, (x_4)$

Figure 4.2: An illustration of the loss functions f_t for the OCO with finite memory lower bound. The teal and orange shading indicate whether or not a decision was used to construct the loss function in a round. For example, in round 5, the decisions x_3 and x_4 are shaded teal because they are used to construct the loss function f_5. However, the decision x_5 is shaded orange because it is not used to construct f_5.

Specialization to OCO with ρ-discounted infinite memory. Now we show how our bounds specialize to the special case of OCO with ρ-discounted infinite memory (Section 4.2.3). For simplicity, we consider the case when the problem follows linear sequence dynamics with the 2-norm instead of a general p-norm.

Theorem 4.6. *Consider an online convex optimization with ρ-discounted infinite memory problem $(\mathcal{X}, \mathcal{H}, A_{\mathit{infinite},\rho}, B_{\mathit{infinite}})$. Suppose that the problem follows linear sequence dynamics with the 2-norm. Let the regularizer $R : \mathcal{X} \to \mathbb{R}$ be α-strongly-convex and satisfy $|R(x) - R(\tilde{x})| \leq D$ for all $x, \tilde{x} \in \mathcal{X}$. Algorithm 4 with step-size η satisfies*

$$\mathrm{Reg}_T^{\mathrm{OCO-UM}}(\mathrm{FTRL}) \leq O\left(\sqrt{\frac{D}{\alpha} T L \tilde{L} (1-\rho^2)^{-\frac{3}{2}}}\right) \leq O\left(\sqrt{\frac{D}{\alpha} T L^2 (1-\rho)^{-2}}\right).$$

Theorem 4.7. *Let $\rho \in [\frac{1}{2}, 1)$. There exists an instance of the online convex optimization with ρ-discounted infinite memory problem, $(\mathcal{X}, \mathcal{H}, A_{infinite,\rho}, B_{infinite})$, that follows linear sequence dynamics with the 2-norm and there exist L-Lipschitz continuous loss functions $\{f_t : \mathcal{H} \to \mathbb{R}\}_{t=1}^{T}$ such that the regret of any algorithm $\mathcal{A}$ satisfies*

$$\mathsf{Reg}_T^{\mathsf{OCO-UM}}(\mathcal{A}) \geq \Omega\left(\sqrt{TL^2(1-\rho)^{-2}}\right).$$

Comparison of upper bound with prior work. The algorithmic ideas and analysis for our regret upper bound are influenced by Anava et al. [2015]. However, an important innovation in our work is the use of weighted norms in the case of linear sequence dynamics. This is a simple but powerful way of encoding prior knowledge about a problem, and allows us to derive non-trivial regret bounds in the case of unbounded-length histories. The technical complications that arise are captured in bounding the relevant quantities of interest, e.g., the Lipschitz constant $\tilde{L}$, the operator norm $\|A^k\|$, etc. Furthermore, using weighted norms even leads to improved regret bounds for some applications. Indeed, consider the application to online linear control with adversarial disturbances (Section 4.4.1). Our framework and upper bound applied to this problem (Theorem 4.8) improve upon the existing upper bound, which used a finite memory approximation. See Lemmas 4.12.2 and 4.12.6 for an illustration of the technical details involved when using weighted norms.

4.4 Applications

In this section we apply our framework to online linear control (Section 4.4.1) and online performative prediction (Section 4.4.2). We defer expanded details and proofs to Sections 4.12 and 4.13 respectively.

4.4.1 Online Linear Control

Background. Online linear control (OLC) is the problem of controlling a system with linear dynamics, adversarial disturbances, and adversarial and convex losses. It combines aspects from control theory and online learning. We refer the reader to Agarwal et al. [2019b] for more details. Here, we introduce the basic mathematical setup of the problem.

Let $\mathcal{S} \subseteq \mathbb{R}^{d_s}$ and $\mathcal{U} \subseteq \mathbb{R}^{d_u}$ denote the state and control spaces. Let s_t and u_t denote the state and control at time t with s_1 being the inital state. The system evolves according to the linear dynamics

$$s_{t+1} = Fs_t + Gu_t + w_t,$$

where $F \in \mathbb{R}^{d_s \times d_s}, G \in \mathbb{R}^{d_s \times d_u}$ are matrices satisfying $\|F\|_2, \|G\|_2 \leq \kappa$ and $w_t \in \mathbb{R}^{d_s}$ is an adversarially chosen disturbance with $\|w_t\|_2 \leq W$. Without loss of generality, we assume that $\kappa, W \geq 1$, $d_s = d_u = d$, and also define $w_0 = s_1$. For $t \in \{1, \ldots, T\}$, let $c_t : \mathcal{S} \times \mathcal{U} \to [0, 1]$ be convex loss functions chosen by an oblivous adversary. The functions c_t satisfy the following Lipschitz condition:

$$\|s\|_2, \|u\|_2 \leq D_{\mathcal{X}} \Rightarrow \|\nabla_s c_t(s,u)\|, \|\nabla_u c_t(s,u)\| \leq L_0 D_{\mathcal{X}}.$$

The goal in online linear control is to choose a sequence of policies that yield a sequence of controls u_t to minimize the regret

$$\mathsf{Reg}_T^{\mathsf{OLC}}(\mathsf{ALG}) = \sum_{t=1}^{T} c_t(s_t, u_t) - \min_{\pi^* \in \Pi} \sum_{t=1}^{T} c_t(s_t^{\pi^*}, u_t^{\pi^*}),$$

where s_t evolves according to linear dynamics stated above, Π denotes a controller class, and $s_t^{\pi^*}, c_t^{\pi^*}$ denote the state and control at time t when the controls are chosen according to π^*.

A very simple controller class is constant input, i.e., $\Pi = \{\pi_u : \pi(s) = u \in \mathcal{U}\}$. In this case, the history h_t can be represented by the finite-dimensional state s_t, and the operators can be set to $A = F$ and $B = \frac{G}{\|G\|}$. However, like previous work [Agarwal et al., 2019b] we focus on the class of (κ, ρ)-strongly stable linear controllers

$$\mathcal{K} = \{K \in \mathbb{R}^{d \times d} : F - GK = HLH^{-1} \text{ s.t. } \|K\|_2, \|H\|_2, \|H^{-1}\|_2 \leq \kappa \text{ and } \|L\|_2 = \rho < 1\}.$$

Given a controller $K \in \mathcal{K}$, the inputs are chosen as linear functions of the current state, i.e., $u_t = -Ks_t$. Unfortunately, parameterizing u_t directly with a linear controller as $u_t = -Ks_t$ leads to a non-convex problem because s_t is a non-linear function of K, e.g., if disturbances are 0, then $s_t = (F - GK)^{t-1} s_0$. An alternative parameterization is the disturbance-action controller (DAC).

Definition 4.4.1. Let $K \in \mathcal{K}$ be fixed. The class of disturbance-action controllers (DACs) is defined as

$$\mathcal{M}_K = \{(K, M) : M = (M^{[s]})_{s=1}^{\infty} \text{ s.t. } M^{[s]} \in \mathbb{R}^{d \times d} \text{ and } \|M^{[s]}\|_2 \leq \kappa^4 \rho^s\}.$$

The control in round k is chosen as

$$u_k = -K s_k + \sum_{j=1}^{k} M^{[j]} w_{k-j}.$$

The class of such DACs has two important properties. First, it acts on the entire history of past disturbances. Consequently, given an arbitrary $K \in \mathcal{K}$, every $K^* \in \mathcal{K}$ can be expressed as a DAC $(K, M) \in \mathcal{M}_K$ with $M = (M^{[1]}, \dots, M^{[T]}, 0, \dots)$ [Agarwal et al., 2019a, Section 16.5]. That is, $\mathcal{K} \subseteq \mathcal{M}_K$ and it suffices to compute regret against $\mathcal{M}_K$ instead of $\mathcal{K}$. For the rest of this chapter we fix $K \in \mathcal{K}$ and denote $\widetilde{F} = F - GK$. Second, suppose $M_t = (M_t^{[s]})_{s=1}^{\infty}$ is the parameter chosen in round t and the control u_t is chosen according to the DAC (K, M_t). Then, s_t and u_t are *linear* functions of the parameters, which implies that c_t is convex in the parameters. (See the next paragraph on "Formulation as OCO with Unbounded Memory" for a formula.) A similar parameterization was first considered for online linear control by Agarwal et al. [2019b] and is based on similar ideas in control theory, e.g., Youla [Youla et al., 1976, Kučera, 1975] and SLS [Wang et al., 2019, Anderson et al., 2019].

Formulation as OCO with Unbounded Memory. Now we formulate the online linear control problem in our framework by defining the decision space $\mathcal{X}$, the history space $\mathcal{H}$, and the linear operators $A : \mathcal{H} \to \mathcal{H}$ and $B : \mathcal{W} \to \mathcal{H}$. Then, we define the functions $f_t : \mathcal{H} \to \mathbb{R}$ in terms of c_t and finally, prove an upper bound on the policy regret. For notational convenience, let $(M^{[s]})$ and (Y_k) denote the sequences $(M^{[1]}, M^{[2]}, \dots)$ and $(Y_0, Y_1, \dots)$ respectively.

Recall that we fix $K \in \mathcal{K}$ to be an arbitrary (κ, ρ)-strongly stable linear controller and consider the disturbance-action controller policy class $\mathcal{M}_K$ (Definition 4.4.1). For the rest of this chapter let $\widetilde{F} = F - GK$. The first step is a change of variables with respect to the control inputs from linear controllers to DACs and the second is a corresponding change of variables for the state. Define the decision space $\mathcal{X}$ as

$$\mathcal{X} = \{M = (M^{[s]}) : M^{[s]} \in \mathbb{R}^{d \times d}, \|M^{[s]}\|_2 \leq \kappa^4 \rho^s\} \tag{4.2}$$

with

$$\|M\|_{\mathcal{X}} = \sqrt{\sum_{j=1}^{\infty} \rho^{-j} \|M^{[j]}\|_F^2}. \tag{4.3}$$

Define the history space $\mathcal{H}$ to be the set consisting of sequences $h = (Y_k)$, where $Y_0 \in \mathcal{X}$ and $Y_k = \widetilde{F}^{k-1} G X_k$ for $X_k \in \mathcal{X}, k \geq 1$ with

$$\|h\|_{\mathcal{H}} = \sqrt{\sum_{k=0}^{\infty} \xi_k^2 \|Y_k\|_{\mathcal{X}}^2}, \tag{4.4}$$

where the weights (ξ_k) are nonnegative real numbers defined as

$$\xi = (1, 1, 1, \rho^{-\frac{1}{2}}, \rho^{-1}, \rho^{-\frac{3}{2}}, \dots). \tag{4.5}$$

Define the linear operators $A : \mathcal{H} \to \mathcal{H}$ and $B : \mathcal{W} \to \mathcal{H}$ as

$$A((Y_0, Y_1, \dots)) = (0, GY_0, \widetilde{F}Y_1, \widetilde{F}Y_2, \dots) \quad \text{and} \quad B(M) = (M, 0, 0, \dots).$$

Note that the problem follows linear sequence dynamics with the ξ-weighted 2-norm (Definition 4.2.3), where ξ is defined above in Eq. (4.5). The weights in the weighted norms on $\mathcal{X}$ and $\mathcal{H}$ increase exponentially. However, the norms $\|M^{[s]}\|_F^2$ and $\|\widetilde{F}^{k-1}G\|_F^2$ decrease exponentially as well: by definition of $M^{[s]}$ in Eq. (4.3) and the assumption on $\widetilde{F} = F - GK$ for $K \in \mathcal{K}$. Leveraging this exponential decrease in $\|M^{[s]}\|_F^2$ and $\|\widetilde{F}^{k-1}G\|_F^2$ to define exponentially increasing weights turns out to be crucial for deriving our regret bounds that are stronger than existing results. Furthermore, the choice to have $\xi_p = 1$ for $p \in \{1, 2\}$ in addition to $p = 0$ (as required by Definition 4.2.3) might seem like a small detail, but this also turns out to be crucial for avoiding unnecessary factors of ρ^{-1} in the regret bounds.

Recall that the loss functions in the online linear control problem are $c_t(s_t, u_t)$, where s_t and u_t are the state and control at round t. Now we will show how to construct the functions $f_t : \mathcal{H} \to \mathbb{R}$ that correspond to $c_t(s_t, u_t)$. By definition, given a sequence of decisions $(M_1, \dots, M_t)$, the history at the end of round t is given by

$$h_t = (M_t, GM_{t-1}, \widetilde{F}GM_{t-2}, \dots, \widetilde{F}^{t-2}GM_0, 0, \dots).$$

A simple inductive argument shows that the state and control in round t can be written as

$$s_t = \widetilde{F}^{t-1}s_1 + \sum_{k=1}^{t-1}\sum_{j=1}^{k} \widetilde{F}^{t-k-1}GM_k^{[j]}w_{k-j} + w_{t-1}, \tag{4.6}$$

$$u_t = -Ks_t + \sum_{j=1}^{t} M_t^{[j]}w_{t-j}. \tag{4.7}$$

Define the functions $f_t : \mathcal{H} \to \mathbb{R}$ by $f_t(h) = c_t(s, u)$, where s and u are the state and control determined by the history as above. Note that f_t is parameterized by the past disturbances. Since the state and control are linear functions of the history and c_t is convex, this implies that f_t is convex.

With the above formulation and the fact that the class of disturbance-action controllers is a superset of the class of (κ, ρ)-strongly-stable linear controllers, we have that the policy regret for the online linear control problem is at most

$$\mathsf{Reg}_T^{\mathsf{OLC}}(\mathsf{ALG}) = \sum_{t=1}^{T} f_t(h_t) - \min_{M \in \mathcal{X}} \sum_{t=1}^{T} \tilde{f}_t(M).$$

This completes the specification of the online convex optimization with unbounded memory problem, $(\mathcal{X}, \mathcal{H}, A, B)$, corresponding to the online linear control problem. Using Algorithm 4 and Theorem 4.2 we can upper bound the above by

$$\mathsf{Reg}_T^{\mathsf{OLC}}(\mathsf{FTRL}) = O\left(\sqrt{\frac{D}{\alpha} T L \tilde{L} H_2}\right),$$

where L is the Lipschitz constant of f_t, $\tilde{L}$ is the Lipschitz constant of $\tilde{f}_t$, H_2 is the 2-effective memory capacity, and $D = \max_{x, \tilde{x} \in \mathcal{X}} |R(x) - R(\tilde{x})|$ for an α-strongly-convex regularizer $R : \mathcal{X} \to \mathbb{R}$. Next, we bound these quantities in terms of the problem parameters of the online linear control problem. We use $O(\cdot)$ to hide absolute constants. The following is our main result for online linear control and it improves existing results [Agarwal et al., 2019b] by a factor of $O(d \log(T)^{3.5} \kappa^5 (1-\rho)^{-1})$. We present a detailed comparison after the proofs in Section 4.12. (See this paragraph.)

Theorem 4.8. *Consider the online linear control problem as defined in Section 4.4.1. Suppose the decisions in round t are chosen using Algorithm 4. Then, the upper bound on* *the policy regret is*

$$\mathsf{Reg}_T^{\mathsf{OLC}}(\mathsf{FTRL}) = O\left(L_0 W^2 \sqrt{T} d^{\frac{1}{2}} \kappa^{17} (1-\rho)^{-4.5}\right). \tag{4.8}$$

Comparison with prior and concurrent work. Existing works solve OLC (and its extensions) by making multiple finite memory approximations. First, they formulate the problem as OCO with finite memory. This requires bounding numerous error terms because the problem is inherently an OCO with unbounded memory problem. We bypass these error analysis steps entirely because the problem fits into our framework naturally. Second, existing works use the parameterization from Agarwal et al. [2019b] that only acts on a fixed, constant number of past disturbances. In particular, existing works use a "truncated" DAC policy that is a sequence of $d \times d$ matrices of length $2\kappa^4(1-\rho)^{-1}\log T$. Our DAC policy acts on the entire history of disturbances and is a sequence of $d \times d$ matrices of unbounded length. Yet, we capture the dimension of this infinite-dimensional space in a way that still improves the overall bound, including completely eliminating the dependence on $\log T$, and improving the dependence on d, κ, and $(1-\rho)$. This improvement comes from our novel use of weighted norms on the history and decision spaces. These norms allow us to give tighter bounds on the relevant quantities in the regret upper bound, e.g., $\|A^k\|$ (Lemma 4.12.2) and $\tilde{L}$ (Lemma 4.12.6).

In complementary concurrent work, Lin et al. [2023] focus on a more general online control problem. They improve regret bounds for this general version by a factor of $\log T$ compared to existing reductions to OCO with finite memory. They do so by using that the impact of a past policy decays geometrically with time. On the other hand, the primary focus of our work is studying the complete dependence of present losses on the entire history in OCO. Applying our resulting OCO with unbounded memory framework to OLC, we improve upon existing results for OLC by removing all $\log T$ factors and improving the dependence on d, κ, and $(1-\rho)$.

4.4.2 Online Performative Prediction

Background. In many applications of machine learning the algorithm's decisions influence the data distribution, e.g., online labor markets [Anagnostopoulos et al., 2018, Horton, 2010], predictive policing [Lum and Isaac, 2016], on-street parking [Dowling et al., 2020, Pierce and Shoup, 2018], vehicle sharing markets [Banerjee et al., 2015], etc. Motivated by such applications, several works have studied the problem of performative prediction, which models the data distribution as a function of the decision-maker's decision [Perdomo et al., 2020, Mendler-Dünner et al., 2020, Miller et al., 2021, Brown et al., 2022, Ray et al., 2022, Jagadeesan et al., 2022]. Most of these works view the problem as a stochastic optimization problem; Jagadeesan et al. [2022] adopt a regret minimization perspective. We refer the reader to these citations for more details. As a natural extension to existing works, we introduce an online learning variant of performative prediction (OPP) with geometric decay [Ray et al., 2022] that differs from the original formulations in a few key ways.

Let the decision set $\mathcal{X} \subseteq \mathbb{R}^d$ be closed and convex with $\|x\|_2 \leq D_{\mathcal{X}}$. Let p_1 denote the initial data distribution over the instance space $\mathcal{Z}$. In each round $t \in [T]$, the learner chooses a decision $x_t \in \mathcal{X}$ and an oblivious adversary chooses a loss function $l_t : \mathcal{X} \times \mathcal{Z} \to [0, 1]$, and then the learner suffers the loss $L_t(x_t) = \mathbb{E}_{z \sim p_t} [l_t(x_t, z)]$, where $p_t = p_t(x_1, \ldots, x_t)$ is the data distribution in round t. The goal in our online learning setting is to minimize the difference between the algorithm's total loss and the total loss of the best fixed decision, $\sum_{t=1}^{T} \mathbb{E}_{z \sim p_t} [l_t(x_t, z)] - \min_{x \in \mathcal{X}} \sum_{t=1}^{T} \mathbb{E}_{z \sim p_t(x)} [l_t(x, z)]$, where $p_t(x) = p_t(x, \ldots, x)$ is the data distribution in round t had x been chosen in all rounds so far. This measure is similar to performative regret [Jagadeesan et al., 2022] and is a natural generalization of performative optimality [Perdomo et al., 2020] for an online learning formulation.

We make the following assumptions. First, the loss functions l_t are convex and L_0-Lipschitz continuous. Second, the data distribution satisfies for all $t \geq 1$, $p_{t+1} = \rho p_t + (1 - \rho)\mathcal{D}(x_t)$, where $\rho \in (0, 1)$ and $\mathcal{D}(x_t)$ is a distribution over $\mathcal{Z}$ that depends on the decision x_t [Ray et al., 2022]. Third, $\mathcal{D}(x)$ is a location-scale distribution: $z \sim \mathcal{D}(x)$ iff $z \sim \xi + Fx$, where $F \in \mathbb{R}^{d \times d}$ satisfies $\|F\|_2 < \infty$ and ξ is a random variable with mean μ and covariance Σ [Ray et al., 2022].

Our problem formulation differs from existing work in the following ways. First, we adopt an online learning perspective on performative prediction with geometric decay, whereas Ray et al. [2022] adopt a stochastic optimization one. So, we assume that the loss functions l_t are adversarially chosen, whereas Ray et al. [2022] assume $l_t = l$ are fixed. Second, we assume that the dynamics ($\mathcal{D}$ and ρ) are known (Assumption A1), whereas Ray et al. [2022] assume they are unknown and use samples from the data distribution. We believe that an appropriate extension of our framework that can deal with unknown linear operators A and B can be applied to this more difficult setting, and we leave this as future work. Third, even though Jagadeesan et al. [2022] also study an online learning variant of performative prediction, they assume $l_t = l$ are fixed and the data distribution depends only on the current decisions, whereas we assume the data distribution depends on the entire history of decisions.

Formulation as OCO with Unbounded Memory. Now we formulate the online performative prediction problem in our framework by defining the decision space $\mathcal{X}$, the history space $\mathcal{H}$, and the linear operators $A : \mathcal{H} \to \mathcal{H}$ and $B : \mathcal{W} \to \mathcal{H}$. Then, we define the functions $f_t : \mathcal{H} \to \mathbb{R}$ in terms of l_t and finally, prove an upper bound on the policy regret. For notational convenience, let (y_k) denote the sequence $(y_0, y_1, \dots)$.

Let $\rho \in (0, 1)$. Let the decision space $\mathcal{X} \subseteq \mathbb{R}^d$ be closed and convex with $\|\cdot\|_{\mathcal{X}} = \|\cdot\|_2$. Let the history space $\mathcal{H}$ be the ℓ^1-direct sum of countably infinte number of copies of $\mathcal{X}$. Define the linear operators $A : \mathcal{H} \to \mathcal{H}$ and $B : \mathcal{X} \to \mathcal{H}$ as

$$A((y_0, y_1, \dots)) = (0, \rho y_0, \rho y_1, \dots) \quad \text{and} \quad B(x) = (x, 0, \dots).$$

Note that the problem is an OCO with ρ-discounted infinite memory problem and follows linear sequence dynamics with the 1-norm (Definition 4.2.3).

Given a sequence of decisions $(x_k)_{k=1}^t$, the history is $h_t = (x_t, \rho x_{t-1}, \dots, \rho^{t-1} x_1, 0, \dots)$ and the data distribution $p_t = p_t(h_t)$ satisfies:

$$z \sim p_t \text{ iff } z \sim \sum_{k=1}^{t-1} (1-\rho)\rho^{k-1}(\xi + F x_{t-k}) + \rho^t p_1. \tag{4.9}$$

This follows from the recursive definition of p_t and parametric assumption about $\mathcal{D}(x)$. Define the functions $f_t : \mathcal{H} \to [0, 1]$ by

$$f_t(h_t) = \mathbb{E}_{z \sim p_t}[l_t(x_t, z)].$$

With the above formulation and definition of f_t, the original goal of minimizing the difference between the algorithm's total loss and the total loss of the best fixed decision is equivalent to minimizing the policy regret,

$$\mathsf{Reg}_T^{\mathsf{OPP}}(\mathsf{ALG}) = \sum_{t=1}^T f_t(h_t) - \min_{x \in \mathcal{X}} \sum_{t=1}^T \tilde{f}_t(x).$$

Theorem 4.9. *Consider the online performative prediction problem as defined in Section 4.4.2. Suppose the decisions in round t are chosen using Algorithm 4. Then, the upper bound on the policy regret is*

$$\mathsf{Reg}_T^{\mathsf{OPP}}(\mathsf{FTRL}) = O\left(D_{\mathcal{X}} L_0 \sqrt{T} \|F\|_2 (1-\rho)^{-\frac{1}{2}} \rho^{-1}\right).$$

We defer the detailed proofs to Section 4.13.

4.5 Implementation Details

In this section we discuss how to implement Algorithm 4 efficiently.

Dimensionality of $\mathcal{X}$. First, note that the decisions $x \in \mathcal{X}$ could be high-dimensional, e.g., an unbounded sequence of matrices as in the online linear control problem, but this is external to our framework and is application dependent. Our framework can be applied to $\mathcal{X}$ or to a lower-dimensional decision space $\mathcal{X}'$. However, the choice of $\mathcal{X}'$ and analyzing the difference

$$\min_{x' \in \mathcal{X}'} \sum_{t=1}^{T} \tilde{f}_t(x') - \min_{x \in \mathcal{X}} \sum_{t=1}^{T} \tilde{f}_t(x)$$

is application dependent. For example, for the online linear control problem one could consider a restricted class of disturbance-action controllers that operate on a constant number of past disturbances as opposed to all the past disturbances, and then analyze the difference between these two policy classes. See, for example, Agarwal et al. [2019b, Lemma 5.2].

Computational cost of each iteration of Algorithm 4. Now we discuss how to implement each iteration of Algorithm 4 efficiently. We are interested in the computational cost of computing the decision x_{t+1} as a function of t. (Given the above discussion about the dimensionality of $\mathcal{X}$, we ignore the fact that the dimensionality of the decisions themselves could depend on t.) Therefore, for the purposes of this section we (i) use $O(\cdot)$ notation to hide absolute constants and problem parameters excluding t and T; (ii) invoke the operators A and B by calling oracles $O_A(\cdot)$ and $O_B(\cdot)$; and (iii) evaluate the functions f_t by calling oracles $O_f(t, \cdot)$. Recall from Assumption A1 that we assume the learner knows the operators A and B, and observes f_t at the end of each round t. So, the oracles O_A, O_B, and O_f are readily available.

Algorithm 4 chooses the decision x_{t+1} as

$$x_{t+1} \in \arg\min_{x\in\mathcal{X}} \sum_{s=1}^{t} \tilde{f}_s(x) + \frac{R(x)}{\eta} = \arg\min_{x\in\mathcal{X}} \underbrace{\sum_{s=1}^{t} f_s\left(\sum_{k=0}^{s-1} A^k Bx\right)}_{=f_{1:t}(x)} + \frac{R(x)}{\eta}.$$

Since $f_{1:t}(x)$ is a sum of $f_1, \ldots, f_t$, evaluating $f_{1:t}(x)$ requires $\Theta(t)$ oracle calls to $\mathcal{O}_f$. However, this issue is present in FTRL for OCO and OCO with finite memory as well and is not specific to our framework. To deal with this issue, one could consider mini-batching algorithms [Dekel et al., 2012, Altschuler and Talwar, 2018, Chen et al., 2020] such as Algorithm 5.

A naïve implementation to evaluate $f_{1:t}(x)$ could require $O(t^3)$ oracle calls to $\mathcal{O}_A$: for each $s \in [t]$, constructing the argument $\sum_{k=0}^{s-1} A^k Bx$ for f_s could require k oracle calls to $\mathcal{O}_A$ to compute $A^k Bx$, for a total of $O(s^2)$ oracle calls. However, $f_{1:t}(x)$ can be evaluated with just $O(t)$ oracle calls to $\mathcal{O}_A$ by constructing the arguments incrementally. For $t \geq 0$, define $\Gamma_t : \mathcal{X} \to \mathcal{H}$ as

$$\Gamma_0(x) = Bx$$

$$\Gamma_t(x) = A\left(\Gamma_{t-1}(x)\right) \qquad \text{for } t \geq 1.$$

Note that $\Gamma_t(Bx) = A^t Bx$. Also, for $t \geq 1$, define $\Upsilon_t : \mathcal{X} \to \mathcal{H}$ as

$$\Upsilon_1(x) = \Gamma_0(x)$$

$$\Upsilon_t(x) = \Upsilon_{t-1}(x) + \Gamma_{t-1}(x) \qquad \text{for } t \geq 2.$$

Note that $\Upsilon_s(x) = \sum_{k=0}^{s-1} A^k Bx$ is the argument for f_s. These can be constructed incrementally as follows.

1. Construct $\Gamma_0(x)$ using one oracle call to $\mathcal{O}_B$.

2. For $s = 1$,

   (a) Construct $\Upsilon_1(x) = \Gamma_0(x)$.

   (b) Construct $\Gamma_1(x)$ from $\Gamma_0(x)$ using one oracle call to $\mathcal{O}_A$.

3. For $s \geq 2$,

   (a) Construct $\Upsilon_s(x)$ by adding $\Upsilon_{s-1}(x)$ and $\Gamma_{s-1}(x)$. This can be done in $O(1)$ time. Recall from our earlier discussion that $O(\cdot)$ hides absolute constants and problem parameters excluding t and T.

   (b) Construct $\Gamma_s(x)$ from $\Gamma_{s-1}(x)$ using one oracle call to $\mathcal{O}_A$.

By incrementally constructing $\Upsilon_s(x)$ as above, we can evaluate $f_{1:t}(x)$ in $O(t)$ time with $O(1)$ oracle calls to $\mathcal{O}_B$, $O(t)$ oracle calls to $\mathcal{O}_A$, and $O(t)$ oracle calls to $\mathcal{O}_f$.

Memory usage of Algorithm 4. We end with a brief discussion of the memory usage of Algorithm 4. We are interested in the memory usage of computing the decision x_{t+1} as a function of t. (Given the discussion about the dimensionality of $\mathcal{X}$ at the start of this section, we ignore the fact that the dimensionality of the decisions themselves could depend on t.) For each $t \in [T]$, the memory usage could be as low as $O(1)$ (if, for example, $\mathcal{X} \subseteq \mathbb{R}^d$, and $A, B \in \mathbb{R}^{d \times d}$, which implies that $\Upsilon_t(x)$ is a d-dimensional vector) or as high as $O(t)$ (if, for example, $\Upsilon_t(x)$ is a t-length sequence of d-dimensional vectors). However, the memory usage is already $\Omega(t)$ to store the functions $f_1, \ldots, f_t$. Therefore, Algorithm 4 only incurs a constant factor overhead.

4.6 An Algorithm with A Low Number of Switches: Mini-Batch FTRL

In this section we present an algorithm (Algorithm 5) for OCO with unbounded memory that provides the same upper bound on policy regret as Algorithm 4 while guaranteeting a small number of switches. Algorithm 5 combines FTRL on the functions $\tilde{f}_t$ with a mini-batching approach. First, it divides rounds into batches of size S, where S is a parameter. Second, at the start of batch $b \in \{1, \ldots, \lceil T/S \rceil\}$, it performs FTRL on the functions $\{g_1, \ldots, g_b\}$, where g_i is the average of the functions $\tilde{f}_t$ in batch i. Then, it uses this decision for the entirety of the current batch. By design, Algorithm 5 switches decisions at most $O(T/S)$ times. This algorithm is insipired by similar algorithms for online learning and OCO [Dekel et al., 2012, Altschuler and Talwar, 2018, Chen et al., 2020].

Theorem 4.10. *Consider an online convex optimization with unbounded memory problem specified by $(\mathcal{X}, \mathcal{H}, A, B)$. Let the regularizer $R : \mathcal{X} \to \mathbb{R}$ be α-strongly-convex and satisfy $|R(x) - R(\tilde{x})| \leq D$ for all $x, \tilde{x} \in \mathcal{X}$. Algorithm 5 with batch size S and step-size η satisfies*

$$\mathsf{Reg}_T^{\mathsf{OCO-UM}}(\mathsf{Mini-Batch\ FTRL}) \leq \frac{SD}{\eta} + \eta \frac{T\tilde{L}^2}{\alpha} + \eta \frac{TL\tilde{L}H_1}{S\alpha}.$$

Algorithm 5: Mini – Batch FTRL

```
Input : Time horizon $T$, step size $\eta$, $\alpha$-strongly-convex regularizer
        $R : \mathcal{X} \to \mathbb{R}$, batch size $S$.
1  Initialize history $h_0 = 0$.
2  for $t = 1, 2, \ldots, T$ do
3      if $t \mod S = 1$ then
4          Let $N_t = \{1, \ldots, \lceil \frac{t}{S} \rceil\}$ denote the number of batches so far.
5          For $b \in N_t$, let $T_b = \{(b-1)S + 1, \ldots, bS\}$ denote the rounds in batch $b$.
6          For $b \in N_t$, let $g_b = \frac{1}{S} \sum_{s \in T_b} \tilde{f}_s$. denote the average of the functions in
             batch $b$.
7          Learner chooses $x_t \in \arg\min_{x \in \mathcal{X}} \sum_{b \in N_t} g_b(x) + \frac{R(x)}{\eta}$.
8      end
9      else
10         Learner chooses $x_t = x_{t-1}$.
11     end
12     Set $h_t = Ah_{t-1} + Bx_t$.
13     Learner suffers loss $f_t(h_t)$ and observes $f_t$.
14 end
```

If $\eta = \sqrt{\frac{\alpha S D}{T \tilde{L} \left(\frac{LH_1}{S} + \tilde{L} \right)}}$, *then*

$$\text{Reg}_T^{\text{OCO-UM}}(\text{Mini – Batch FTRL}) \leq O\left(\sqrt{\frac{D}{\alpha} T \left(L\tilde{L}H_1 + S\tilde{L}^2 \right)} \right).$$

Setting the batch size to be $S = {LH_1}/{\tilde{L}}$ we obtain the same upper bound on policy regret as Algorithm 4 while guaranteeing that the decisions x_t switch at most ${T\tilde{L}}/{LH_1}$ times.

Corollary 4.6.1. *Consider an online convex optimization with unbounded memory problem specified by* $(\mathcal{X}, \mathcal{H}, A, B)$. *Let the regularizer* $R : \mathcal{X} \to \mathbb{R}$ *be* α*-strongly-convex and satisfy* $|R(x) - R(\tilde{x})| \leq D$ *for all* $x, \tilde{x} \in \mathcal{X}$. *Algorithm 5 with batch size* $S = \frac{LH_1}{\tilde{L}}$ *and step-size*

$\eta = \sqrt{\frac{\alpha S D}{T\tilde{L}\left(\frac{LH_1}{S}+\tilde{L}\right)}}$ *satisfies*

$$\text{Reg}_T^{\text{OCO-UM}}(\text{Mini}-\text{Batch FTRL}) \leq O\left(\sqrt{\frac{D}{\alpha}TL\tilde{L}H_1}\right).$$

Furthermore, the decisions x_t switch at most $\frac{T\tilde{L}}{LH_1}$ times.

We defer the proof to Section 4.14. Intuitively, in the OCO with unbounded memory framework each decision x_t is penalized not just in round t but in future rounds as well. Therefore, instead of immediately changing the decision, it is prudent to stick to it for a while, collect more data, and then switch decisions. For the OCO with finite memory problem, the constant memory length m provides a natural measure of how long decisions penalized for and when one should switch decisions. In the general case, this is measured by the quantity ${}^{LH_1}/_{\tilde{L}}$. Note that this simplifies to m for OCO with finite memory for all p-norms.

Note that Theorem 4.10 only provides an upper bound on the policy regret for the general case. Unlike Algorithm 4, it is unclear how to obtain a stronger bound depending on H_p for the case of linear sequence dynamics with the ξ-weighted p-norm for $p > 1$. (See the proofs and discussion in Section 4.14.) However, for the special case of OCO with finite memory, which follows linear sequence dynamics with the 2-norm, we can do so by leveraging the special structure of the linear operator $A_{\text{finite},m}$.

Theorem 4.11. *Consider an online convex optimization with finite memory problem with constant memory length m specified by $(\mathcal{X}, \mathcal{H} = \mathcal{X}^m, A_{finite,m}, B_{finite,m})$. Let the regularizer*

$R : \mathcal{X} \to \mathbb{R}$ be α-strongly-convex and satisfy $|R(x) - R(\tilde{x})| \leq D$ for all $x, \tilde{x} \in \mathcal{X}$. Algorithm 5 with batch size m and step-size $\eta = \sqrt{\frac{\alpha m D}{T\tilde{L}\left(Lm^{\frac{1}{2}} + \tilde{L}\right)}}$ satisfies

$$\mathsf{Reg}_T^{\mathsf{OCO-UM}}(\mathsf{Mini-Batch\ FTRL}) \leq O\left(\sqrt{\frac{D}{\alpha} T L \tilde{L} m^{\frac{3}{2}}}\right) \leq O\left(m\sqrt{\frac{D}{\alpha} T L^2}\right).$$

Furthermore, the decisions x_t switch at most $\frac{T}{m}$ times.

4.7 Experiments

In this section we present some simple simulation experiments. The implementation of our algorithms and code to reproduce the results are available at https://github.com/raunakkmr/oco-with-memory-code.

Problem Setup. We consider the problem of online linear control with a constant input controller class $\Pi = \{\pi_u : \pi(s) = u \in \mathcal{U}\}$. Let T denote the time horizon. Let $\mathcal{S} = \mathbb{R}^d$ and $\mathcal{U} = \{u \in \mathbb{R}^d : \|u\|_2 \leq 1\}$ denote the state and control spaces. Let s_t and u_t denote the state and control at time t with s_0 being the initial state. The system evolves according to linear dynamics $s_{t+1} = Fs_t + Gu_t + w_t$, where $F, G \in \mathbb{R}^{d \times d}$ are system matrices and $w_t \in \mathbb{R}^d$ is a disturbance. The loss function in round t is simply $c_t(s_t, u_t) = c_t(s_t) = \sum_{j=1}^d s_{t,j}$, where $s_{t,j}$ denotes the j-th coordinate of s_t. The goal is to choose a sequence of control inputs $u_0, \ldots, u_{T-1} \in \mathcal{U}$ to minimize the regret

$$\sum_{t=0}^{T-1} c_t(s_t, u_t) - \min_{u \in \mathcal{U}} \sum_{t=0}^{T-1} c_t(s_t^u, u),$$

where s_t^u denotes the state in round t upon choosing control input u in each round. Note that the state in round t can be written as

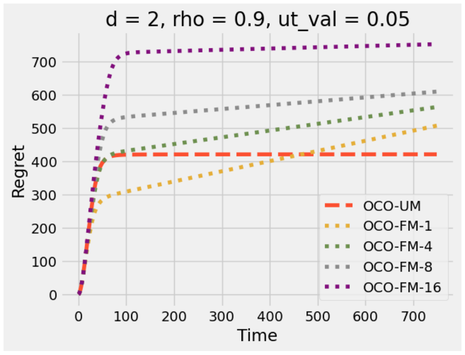
$$s_t = \sum_{k=1}^{t} F^k G u_{t-k} + \sum_{k=1}^{t} F^k w_{t-k}.$$

Therefore, we can formulate this problem as an OCO with unbounded memory problem by setting $\mathcal{X} = \mathcal{U}, \mathcal{H} = \{y \in \mathbb{R}^d : y = \sum_{k=0}^{t} F^k G u \text{ for some } u \in \mathcal{U} \text{ and } t \in \mathbb{N}\}, A(h) = Fh, B(x) = Gx$, and $f_t(h_t) = c_t(\sum_{k=1}^{t} F^k G u_{t-k} + \sum_{k=1}^{t} F^k w_{t-k})$. Note that $\mathcal{H}, A$, and B are all finite-dimensional.

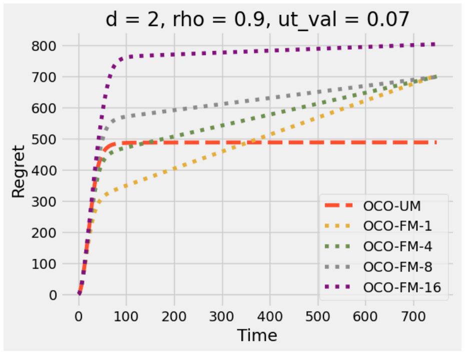
Data. We set the time horizon $T = 750$ and dimension $d = 2$. We sample the disturbances $\{w_t\}$ from a standard normal distribution. We set the system matrix G to be the identity and the system matrix F to be a diagonal plus upper triangular matrix with the diagonal entries equal to ρ and the upper triangular entries equal to α. We run simulations with various values of ρ and α.

Implementation. We use the `cvxpy` library [Diamond and Boyd, 2016, Agrawal et al., 2018] for implementing Algorithm 4. We use step-sizes according to Theorems 4.2 and 4.4. We run the experiments on a standard laptop.

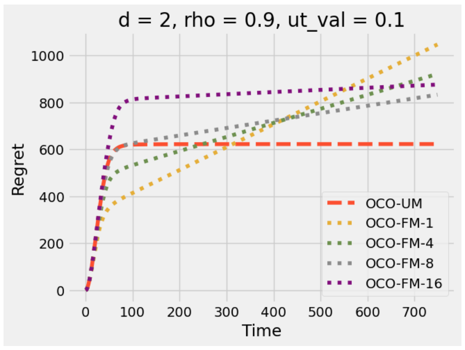
Results. We compare the regret with respect to the optimal control input of OCO with unbounded memory and OCO with finite memory for various memory lengths m in Fig. 4.3 for $\rho = 0.90$ and Fig. 4.4 for $\rho = 0.95$. There are a few important takeaways.



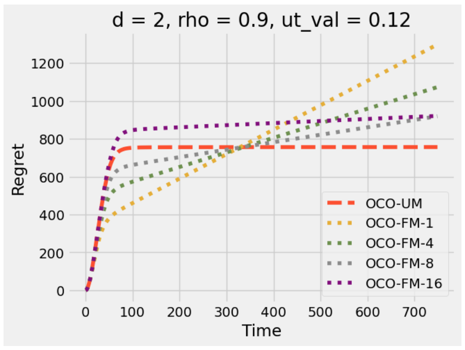
(a)



(b)



(c)



(d)

Figure 4.3: Regret plot for $\rho = 0.90$. The label `OCO-UM` refers to formulating the problem as an OCO with unbounded memory problem. The `OCO-FM-m` refers to formulating the problem as an OCO with finite memory problem with constant memory length m. The titles of the plots indicate the values of the dimension, the diagonal entries of F, and the upper triangular entries of F.

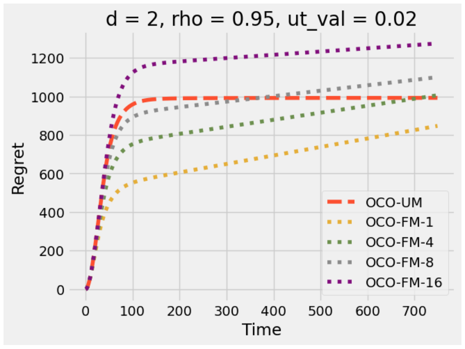
1. OCO with unbounded memory either performs as well as or better than OCO with finite memory, and it does so at comparable computational cost (Section 4.5). In fact, the regret curve for OCO with unbounded memory reaches an asymptote whereas this is not the case for OCO with finite memory for a variety of memory lengths.

2. Knowledge of the spectral radius of F, ρ, is not sufficient to tune the mem-

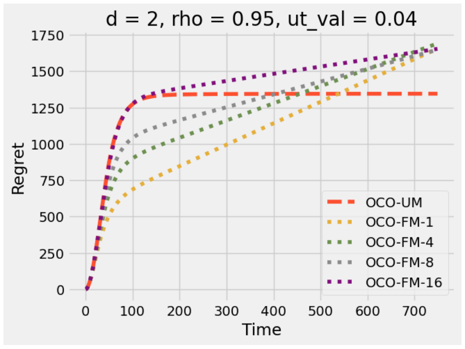
ory length m for OCO with finite memory. This is illustrated by comparing Figs. 4.3a to 4.3d. Even though small memory lengths perform well when the upper triangular value is small, they perform poorly when the upper triangular value is large. In contrast, OCO with unbounded memory performs well in all cases.

3. For a fixed memory length, OCO with unbounded memory eventually performs better than OCO with finite memory. This is illustrated by comparing Figs. 4.3a to 4.3d.

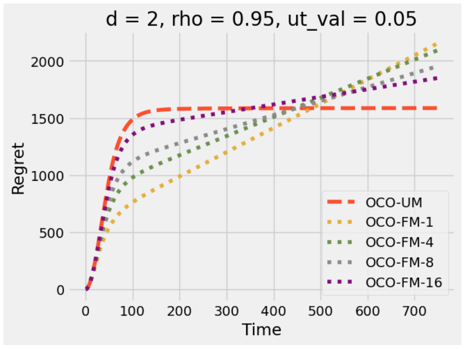
4. As we increase the memory length, the performance of OCO with finite memory eventually approaches that of OCO with unbounded memory. However, an advantage of OCO with unbounded memory is that it does not require tuning the memory length. For example, when $\rho = 0.90$ and the upper triangular entry of $F = 0.10$, OCO with finite memory with $m = 4$ performs comparably to $m = 8$ and $m = 16$ (Fig. 4.3c). However, when the upper triangular entry of $F = 0.12$, then it performs much worse (Fig. 4.3d). However, OCO with unbounded memory performs well in all cases without the need for tuning an additional hyperparameter in the form of memory length.



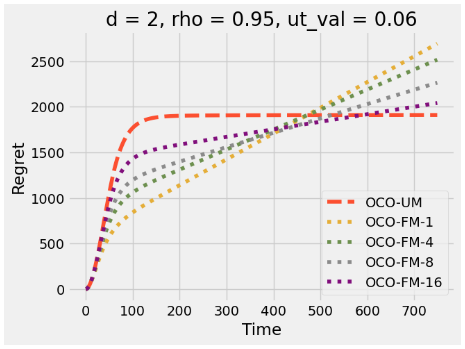
(a)



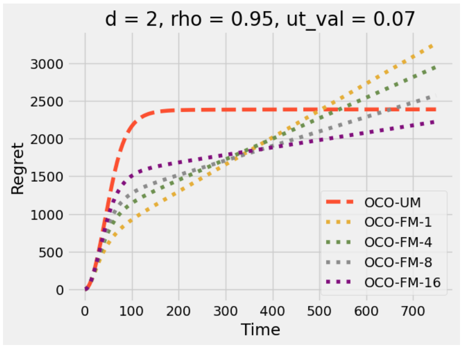
(b)



(c)



(d)



(e)

Figure 4.4: Regret plot for $\rho = 0.95$. The label `OCO-UM` refers to formulating the problem as an OCO with unbounded memory problem. The `OCO-FM-m` refers to formulating the problem as an OCO with finite memory problem with constant memory length m. The titles of the plots indicate the values of the dimension, the diagonal entries of F, and the upper triangular entries of F.

4.8 Discussion

In this chapter we introduced a generalization of the OCO framework, "Online Convex Optimization with Unbounded Memory", that allows the loss in the current round to depend on the entire history of decisions until that point. We proved matching upper and lower bounds on the policy regret in terms of the time horizon, the p-effective memory capacity (a quantitative measure of the influence of past decisions on present losses), and other problem parameters (Theorems 4.2 and 4.3). As a special case, we proved the first non-trivial lower bound for OCO with finite memory (Theorem 4.5), which could be of independent interest, and also improved existing upper bounds (Theorem 4.4). We illustrated the power of our framework by bringing together the regret analysis of two seemingly disparate problems under the same umbrella: online linear control (Theorem 4.8), where we improve and simplify existing regret bounds, and online performative prediction (Theorem 4.9).

There are a number of directions for future research. A natural follow-up is to consider unknown dynamics (i.e., when the learner does not know the operators A and B) and/or the case of bandit feedback (i.e., when the learner only observes $f_t(h_t)$). The extension to bandit feedback has been considered in the OCO and OCO with finite memory literature [Hazan and Li, 2016, Bubeck et al., 2021, Zhao et al., 2021, Gradu et al., 2020, Cassel and Koren, 2020]. It is tempting to think about a version where the history is a *nonlinear*, but decaying, function of the past decisions. The obvious challenge is that the nonlinearity would lead to non-convex losses. It is unclear how to deal with such issues, e.g., restricted classes of nonlinearities for which the OCO with unbounded memory perspective is still relevant [Zhang et al., 2015], different problem formulations such as online nonconvex learning [Gao et al., 2018, Suggala and Netrapalli, 2020], etc.

There is a growing body of work on online linear control and its variants that rely on OCO with finite memory [Hazan et al., 2020, Agarwal et al., 2019c, Foster and Simchowitz, 2020, Cassel and Koren, 2020, Gradu et al., 2020, Li et al., 2021b, Minasyan et al., 2021]. In this chapter we showed how our framework can be used to improve and simplify regret bounds for the online linear control problem. Another direction for future work is to use our framework, perhaps with suitable extensions outlined above, to derive similar improvements for these other variants of online linear control.

4.9 Proofs for Section 4.2: Framework

In this section prove Theorem 4.1. But first we prove a lemma that we use for proofs involving linear sequence dynamics with the ξ-weighted p-norm (Definition 4.2.3). Recall that $\|\cdot\|_{\mathcal{U}}$ denotes the norm associated with a space $\mathcal{U}$ and the operator norm $\|L\|$ for a linear operator $L : \mathcal{U} \to \mathcal{V}$ is defined as $\|L\| = \max_{u:\|u\|_{\mathcal{U}}\leq 1} \|Lu\|_{\mathcal{V}}$.

Lemma 4.9.1. *Consider an online convex optimization with unbounded memory problem specified by $(\mathcal{X}, \mathcal{H}, A, B)$. If $(\mathcal{X}, \mathcal{H}, A, B)$ follows linear sequence dynamics with the ξ-weighted p-norm for $p \geq 1$, then for all $k \geq 1$*

$$\xi_k \|A_{k-1} \cdots A_0\| \leq \|A^k\|.$$

Proof. Let $x \in \mathcal{X}$ with $\|x\|_{\mathcal{X}} = 1$. We have

$$\xi_k \|A_{k-1} \cdots A_0 x\|_{\mathcal{X}} = \|A^k(x, 0, \ldots)\|_{\mathcal{H}} \leq \|A^k\| \|(x, 0, \ldots)\|_{\mathcal{H}} \leq \|A^k\|,$$

where the last inequality follows because $\|(x, 0, \ldots)\|_{\mathcal{H}} = \xi_0 \|x\|_{\mathcal{X}}$ and $\xi_0 = 1$ by Definition 4.2.3. Therefore, $\|A_{k-1} \cdots A_0\| \leq \|A^k\|$. ■

Theorem 4.1. *Consider an online convex optimization with unbounded memory problem specified by $(\mathcal{X}, \mathcal{H}, A, B)$. If f_t is L-Lipschitz continuous,*

$$\tilde{L} \leq L \sum_{k=0}^{\infty} \|A^k\|.$$

If $(\mathcal{X}, \mathcal{H}, A, B)$ follows linear sequence dynamics with the ξ-weighted p-norm for $p \geq 1$, then

$$\tilde{L} \leq L \left(\sum_{k=0}^{\infty} \|A^k\|^p \right)^{\frac{1}{p}}.$$

Proof. Let $x, \tilde{x} \in \mathcal{X}$. For the general case, we have

$$\begin{aligned}
\left|\tilde{f}_t(x) - \tilde{f}_t(\tilde{x})\right| &= \left| f_t\left(\sum_{k=0}^{t-1} A^k B x\right) - f_t\left(\sum_{k=0}^{t-1} A^k B \tilde{x}\right)\right| && \text{by Definition 4.2.1} \\
&\leq L \left\| \sum_{k=0}^{t-1} A^k B(x-\tilde{x}) \right\|_{\mathcal{H}} && f_t \text{ is } L\text{-Lipschitz continuous} \\
&\leq L \sum_{k=0}^{t-1} \|A^k\| \|B\| \|x-\tilde{x}\|_{\mathcal{X}} \\
&\leq L \sum_{k=0}^{t-1} \|A^k\| \|x-\tilde{x}\|_{\mathcal{X}} && \text{by Assumption } \textbf{A2} \\
&\leq L \sum_{k=0}^{\infty} \|A^k\| \|x-\tilde{x}\|_{\mathcal{X}}.
\end{aligned}$$

If $(\mathcal{H}, \mathcal{X}, A, B)$ follows linear sequence dynamics with the ξ-weighted p-norm for $p \geq 1$, then we have

$$\begin{aligned}
\left|\tilde{f}_t(x) - \tilde{f}_t(\tilde{x})\right| &= \left| f_t\left(\sum_{k=0}^{t-1} A^k B x\right) - f_t\left(\sum_{k=0}^{t-1} A^k B \tilde{x}\right)\right| && \text{by Definition 4.2.1} \\
&\leq L \left\| \sum_{k=0}^{t-1} A^k B(x-\tilde{x}) \right\|_{\mathcal{H}} && f_t \text{ is } L\text{-Lipschitz continuous} \\
&= L \; \|(0, A_0(x-\tilde{x}), A_1 A_0(x-\tilde{x}), \ldots)\| && \text{by Definition 4.2.3} \\
&= L \left(\sum_{k=0}^{t-1} \xi_k^p \, \|A_{k-1} \cdots A_0(x-\tilde{x})\|^p \right)^{\frac{1}{p}} && \text{by Definition 4.2.3} \\
&\leq L \left(\sum_{k=0}^{t-1} \|A^k\|^p \right)^{\frac{1}{p}} \|x-\tilde{x}\|_{\mathcal{X}} && \text{by Lemma 4.9.1} \\
&\leq L \left(\sum_{k=0}^{\infty} \|A^k\|^p \right)^{\frac{1}{p}} \|x-\tilde{x}\|_{\mathcal{X}}. \qquad \blacksquare
\end{aligned}$$

4.10 Proofs for Section 4.3: Upper Bounds

First we prove a lemma that bounds the difference in the value of f_t evaluated at the actual history h_t and an idealized history that would have been obtained by playing x_t in all prior rounds.

Lemma 4.10.1. *Consider an online convex optimization with unbounded memory problem specified by* $(\mathcal{X}, \mathcal{H}, A, B)$. *If the decisions* (x_t) *are generated by Algorithm 4, then*

$$\left|f_t(h_t) - \tilde{f}_t(x_t)\right| \leq \eta \frac{L\tilde{L}H_1}{\alpha}$$

for all rounds t. *When* $(\mathcal{X}, \mathcal{H}, A, B)$ *follows linear sequence dynamics with the* ξ*-weighted* p*-norm for* $p \geq 1$, *then*

$$\left|f_t(h_t) - \tilde{f}_t(x_t)\right| \leq \eta \frac{L\tilde{L}H_p}{\alpha}$$

for all rounds t.

Proof. We have

$$\begin{aligned}
\left|f_t(h_t) - \tilde{f}_t(x_t)\right| &= \left|f_t(h_t) - f_t\left(\sum_{k=0}^{t-1} A^k B x_t\right)\right| && \text{by Definition 4.2.1} \\
&\leq L\left\|h_t - \sum_{k=0}^{t-1} A^k B x_t\right\| && \text{by Assumption A4} \\
&= L\left\|\sum_{k=0}^{t-1} A^k B x_{t-k} - \sum_{k=0}^{t-1} A^k B x_t\right\| && \text{by definition of } h_t \\
&= L\underbrace{\left\|\sum_{k=0}^{t-1} A^k B (x_{t-k} - x_t)\right\|}_{(a)}. && (4.10)
\end{aligned}$$

First consider the general case where $(\mathcal{X}, \mathcal{H}, A, B)$ does not necessarily follow linear sequence dynamics. We can bound the term (a) as

$$\begin{aligned}
\left\| \sum_{k=0}^{t-1} A^k B(x_{t-k} - x_t) \right\| &\leq \sum_{k=0}^{t-1} \left\| A^k B \right\| \|x_t - x_{t-k}\| \\
&\leq \sum_{k=0}^{t-1} \left\| A^k B \right\| k\eta \frac{\tilde{L}}{\alpha} && \text{by Theorem 2.1} \\
&\leq \sum_{k=0}^{t-1} \left\| A^k \right\| k\eta \frac{\tilde{L}}{\alpha} && \text{by Assumption } \mathbf{A2} \\
&\leq \eta \frac{\tilde{L}}{\alpha} H_1.
\end{aligned}$$

Plugging this into Eq. (4.10) completes the proof for the general case. Now consider the case when $(\mathcal{X}, \mathcal{H}, A, B)$ follows linear sequence dynamcis with the ξ-weighted p-norm. We can bound the term (a) as

$$\begin{aligned}
\left\| \sum_{k=0}^{t-1} A^k B(x_{t-k} - x_t) \right\| &= \|(0, A_0(x_t - x_{t-1}), A_1 A_0(x_t - x_{t-2}), \ldots)\| && \text{by Definition 4.2.3} \\
&= \left(\sum_{k=0}^{t-1} \xi_k^p \|A_{k-1} \cdots A_0(x_t - x_{t-k})\|^p \right)^{\frac{1}{p}} && \text{by Definition 4.2.3} \\
&\leq \left(\sum_{k=0}^{t-1} \xi_k^p \|A_{k-1} \cdots A_0\|^p \|x_t - x_{t-k}\|^p \right)^{\frac{1}{p}} \\
&\leq \left(\sum_{k=0}^{t-1} \left\| A^k \right\|^p \|x_t - x_{t-k}\|^p \right)^{\frac{1}{p}} && \text{by Lemma 4.9.1} \\
&\leq \eta \frac{\tilde{L}}{\alpha} \left(\sum_{k=0}^{t-1} \left\| A^k \right\|^p k^p \right)^{\frac{1}{p}} && \text{by Theorem 2.1} \\
&\leq \eta \frac{\tilde{L}}{\alpha} H_p.
\end{aligned}$$

Plugging this into Eq. (4.10) completes the proof. ■

Now we restate and prove Theorem 4.2

Theorem 4.2. *Consider an online convex optimization with unbounded memory problem specified by $(\mathcal{X}, \mathcal{H}, A, B)$. Let the regularizer $R : \mathcal{X} \to \mathbb{R}$ be α-strongly-convex and satisfy $|R(x) - R(\tilde{x})| \leq D$ for all $x, \tilde{x} \in \mathcal{X}$. Algorithm 4 with step-size η satisfies*

$$\mathsf{Reg}_T^{\mathsf{OCO-UM}}(\mathsf{FTRL}) \leq \frac{D}{\eta} + \eta \frac{T\tilde{L}^2}{\alpha} + \eta \frac{TL\tilde{L}H_1}{\alpha}.$$

If $\eta = \sqrt{\frac{\alpha D}{T\tilde{L}(LH_1 + \tilde{L})}}$, then

$$\mathsf{Reg}_T^{\mathsf{OCO-UM}}(\mathsf{FTRL}) \leq O\left(\sqrt{\frac{D}{\alpha} TL\tilde{L}H_1}\right).$$

When $(\mathcal{X}, \mathcal{H}, A, B)$ follows linear sequence dynamics with the ξ-weighted p-norm, then all of the above hold with H_p instead of H_1.

Proof. First consider the general case where $(\mathcal{X}, \mathcal{H}, A, B)$ does not necessarily follow linear sequence dynamics. Let $x^* \in \arg\min_{x \in \mathcal{X}} \sum_{t=1}^T \tilde{f}_t(x)$. Note that we can write the regret as

$$\begin{aligned}
\mathsf{Reg}_T(\mathsf{FTRL}) &= \sum_{t=1}^T f_t(h_t) - \min_{x \in \mathcal{X}} \sum_{t=1}^T \tilde{f}_t(x) \\
&= \underbrace{\sum_{t=1}^T f_t(h_t) - \tilde{f}_t(x_t)}_{(a)} + \underbrace{\sum_{t=1}^T \tilde{f}_t(x_t) - \tilde{f}_t(x^*)}_{(b)}.
\end{aligned}$$

We can bound term (a) using Lemma 4.10.1 and term (b) using Theorem 2.1.

Therefore, we have

$$\begin{aligned}\mathsf{Reg}_T(\mathsf{FTRL}) &= \underbrace{\sum_{t=1}^{T} f_t(h_t) - \tilde{f}_t(x_t)}_{(a)} + \underbrace{\sum_{t=1}^{T} \tilde{f}_t(x_t) - \tilde{f}_t(x^*)}_{(b)} \\ &\leq \eta\frac{TL\tilde{L}H_1}{\alpha} + \frac{D}{\eta} + \eta\frac{T\tilde{L}^2}{\alpha}.\end{aligned}$$

Choosing $\eta = \sqrt{\frac{\alpha D}{T\tilde{L}(LH_1+\tilde{L})}}$ yields

$$\mathsf{Reg}_T(\mathsf{FTRL}) \leq O\left(\sqrt{\frac{D}{\alpha}TL\tilde{L}H_1}\right),$$

where we used the definition of p-effective memory capacity (Definition 4.2.4) and the bound on $\tilde{L}$ (Theorem 4.1) to simplify the above expression. This completes the proof for the general case. The proof for when $(\mathcal{X}, \mathcal{H}, A, B)$ follows linear sequence dynamcis with the ξ-weighted p-norm is the same as above, except we bound the term (a) above using Lemma 4.10.1 for linear sequence dynamics. ∎

Now we restate and prove Theorem 4.4.

Theorem 4.4. *Consider an online convex optimization with finite memory problem with constant memory length m specified by $(\mathcal{X}, \mathcal{H} = \mathcal{X}^m, A_{\text{finite},m}, B_{\text{finite},m})$. Let the regularizer $R : \mathcal{X} \to \mathbb{R}$ be α-strongly-convex and satisfy $|R(x)-R(\tilde{x})| \leq D$ for all $x, \tilde{x} \in \mathcal{X}$. Algorithm 4 with step-size $\eta = \sqrt{\frac{\alpha D}{T\tilde{L}(Lm^{\frac{3}{2}}+\tilde{L})}}$ satisfies*

$$\mathsf{Reg}_T^{\mathsf{OCO-UM}}(\mathsf{FTRL}) \leq O\left(\sqrt{\frac{D}{\alpha}TL\tilde{L}m^{\frac{3}{2}}}\right) \leq O\left(m\sqrt{\frac{D}{\alpha}TL^2}\right).$$

The OCO with finite memory problem, as defined in the literature, follows linear sequence dynamics with the 2-norm. Here, we consider a more general version of the OCO with finite memory problem that follows linear sequence dynamics with the p-norm. We provide an upper bound on the policy regret for this more general formulation and the proof of Theorem 4.4 follows as a special case when $p = 2$.

Theorem 4.12. *Consider an online convex optimization with finite memory problem with constant memory length m, $(\mathcal{X}, \mathcal{H} = \mathcal{X}^m, A_{\text{finite},m}, B_{\text{finite},m})$. Assume that the problem follows linear sequence dynamics with the p-norm for $p \geq 1$. Let the regularizer $R : \mathcal{X} \to \mathbb{R}$ be α-strongly-convex and satisfy $|R(x) - R(\tilde{x})| \leq D$ for all $x, \tilde{x} \in \mathcal{X}$. Algorithm 4 with step-size η satisfies*

$$\mathsf{Reg}_T(\mathsf{FTRL}) \leq O\left(\sqrt{\frac{D}{\alpha} T L \tilde{L} m^{\frac{p+1}{p}}}\right) \leq O\left(\sqrt{\frac{D}{\alpha} T L^2 m^{\frac{p+2}{p}}}\right).$$

Proof. Using Theorem 4.2 it suffices to bound $\tilde{L}$ and H_p for this problem. Note that $\|A^k_{\text{finite}}\| = 1$ if $k \leq m$ and 0 otherwise. Using this we have

$$H_p = \left(\sum_{k=0}^{\infty} \left(k \|A^k_{\text{finite}}\|\right)^p\right)^{\frac{1}{p}} = \left(\sum_{k=0}^{m} k^p\right)^{\frac{1}{p}} \leq O\left(m^{\frac{p+1}{p}}\right).$$

This proves the first inequality in the statement of the theorem. The second inequality follows from the above and Theorem 4.1, which states that

$$\tilde{L} \leq L \left(\sum_{k=0}^{\infty} \|A^k_{\text{finite}}\|^p\right)^{\frac{1}{p}} = L m^{\frac{1}{p}}. \qquad \blacksquare$$

Now we provide an upper bound on the policy regret for the OCO with ρ-discounted infinite memory problem. We restate and prove Theorem 4.6

Theorem 4.6. *Consider an online convex optimization with ρ-discounted infinite memory problem $(\mathcal{X}, \mathcal{H}, A_{infinite,\rho}, B_{infinite})$. Suppose that the problem follows linear sequence dynamics with the 2-norm. Let the regularizer $R : \mathcal{X} \to \mathbb{R}$ be α-strongly-convex and satisfy $|R(x) - R(\tilde{x})| \le D$ for all $x, \tilde{x} \in \mathcal{X}$. Algorithm 4 with step-size η satisfies*

$$\mathsf{Reg}_T^{\mathsf{OCO-UM}}(\mathsf{FTRL}) \le O\left(\sqrt{\frac{D}{\alpha} TL\tilde{L}(1-\rho^2)^{-\frac{3}{2}}}\right) \le O\left(\sqrt{\frac{D}{\alpha} TL^2(1-\rho)^{-2}}\right).$$

Proof. Using Theorem 4.2, it suffices to bound $\tilde{L}$ and H_p for this problem. Recall that $\|A^k_{\text{infinite},\rho}\| = \rho^k$. Using this we have

$$H_2 = \left(\sum_{k=0}^{\infty} \left(k\|A^k_{\text{finite}}\|\right)^2\right)^{\frac{1}{2}} = \left(\sum_{k=0}^{\infty} \left(k\rho^k\right)^2\right)^{\frac{1}{2}} \le (1-\rho^2)^{-\frac{3}{2}}.$$

This proves the first inequality in the statement of the theorem. The second inequality follows from the following. Note that Theorem 4.1 yields

$$\tilde{L} \le L\left(\sum_{k=0}^{\infty} \|A^k_{\text{infinite},\rho}\|^2\right)^{\frac{1}{2}} = L(1-\rho^2)^{-\frac{1}{2}}.$$

Therefore,

$$\mathsf{Reg}_T^{\mathsf{OCO-UM}}(\mathsf{FTRL}) \le O\left(\sqrt{\frac{D}{\alpha} TL^2(1-\rho^2)^{-2}}\right).$$

Now, the second inequality follows from using $1-\rho^2 = (1+\rho)(1-\rho)$, which implies that $1-\rho \le 1-\rho^2 \le 2(1-\rho)$ because $\rho \in (0,1)$. ■

Existing Regret Bound for OCO with Finite Memory. Now we provide a detailed comparison of our upper bound on the policy regret for OCO with finite memory with that of Anava et al. [2015]. The material in this subsection comes from Appendix A.2 of their arXiv version or Appendix C.2 of their conference version.

The existing upper bound on regret is

$$O\left(\sqrt{DT\lambda m^{\frac{3}{2}}}\right),$$

where $D = \max_{x,\tilde{x}\in\mathcal{X}} |R(x) - R(\tilde{x})|$. Although the parameter λ is defined in terms of dual norms of the gradient of $\tilde{f}_t$, it is essentially the Lipschitz-continuity constant for $\tilde{f}_t$: for all $x, \tilde{x} \in \mathcal{X}$,

$$\left|\tilde{f}_t(x) - \tilde{f}_t(\tilde{x})\right| \leq \sqrt{\lambda\alpha}\|x - \tilde{x}\|,$$

where α is the strong-convexity parameter of the regularizer R (or σ in the notation of Anava et al. [2015]). Therefore, the existing regret bound can be rewritten as

$$O\left(\tilde{L}\sqrt{\frac{D}{\alpha}Tm^{\frac{3}{2}}}\right).$$

Our upper bound on the policy regret for OCO with finite memory Theorem 4.4 is

$$O\left(\sqrt{\frac{D}{\alpha}L\tilde{L}Tm^{\frac{3}{2}}}\right).$$

Since $\tilde{L} \leq \sqrt{m}L$ by Theorem 4.1, this leads to an improvement by a factor of $m^{\frac{1}{4}}$.

4.11 Proofs for Section 4.3: Lower Bounds

We first restate Theorems 4.3 and 4.5.

Theorem 4.3. *There exists an instance of the online convex optimization with unbounded memory problem, $(\mathcal{X}, \mathcal{H}, A, B)$, that follows linear sequence dynamics with the ξ-weighted p-norm and there exist L-Lipschitz continuous loss functions $\{f_t : \mathcal{H} \to \mathbb{R}\}_{t=1}^{T}$ such that the regret of any algorithm $\mathcal{A}$ satisfies*

$$\mathsf{Reg}_T^{\mathsf{OCO-UM}}(\mathcal{A}) \geq \Omega\left(\sqrt{TL\tilde{L}H_p}\right).$$

Theorem 4.5. *There exists an instance of the online convex optimization with finite memory problem with constant memory length m, $(\mathcal{X}, \mathcal{H} = \mathcal{X}^m, A_{\mathit{finite},m}, B_{\mathit{finite},m})$, and there exist L-Lipschitz continuous loss functions $\{f_t : \mathcal{H} \to \mathbb{R}\}_{t=1}^{T}$ such that the regret of any algorithm $\mathcal{A}$ satisfies*

$$\mathsf{Reg}_T^{\mathsf{OCO-UM}}(\mathcal{A}) \geq \Omega\left(m\sqrt{TL^2}\right).$$

Theorem 4.3 follows from Theorem 4.5. However, the lower bound is true for a much broader class of problems as we show in this section. We first provide a lower bound for a more general formulation of the OCO with finite memory problem (Theorem 4.13). The proof of Theorem 4.5 follows as a special case when $p = 2$. Then, we provide a lower bound for the OCO with ρ-discounted infinite memory problem (Theorem 4.7).

The OCO with finite memory problem, as defined in the literature, follows linear sequence dynamics with the 2-norm. In this section we consider a more general version of the OCO with finite memory problem that follows linear sequence dynamics with the p-norm. We provide a lower bound on the policy regret for this more general formulation and the proof of Theorem 4.5 follows as a special case when $p = 2$.

Theorem 4.13. *For all $p \geq 1$, there exists an instance of the online convex optimization with finite memory problem with constant memory length m, $(\mathcal{X}, \mathcal{H} = \mathcal{X}^m, A_{\textit{finite},m}, B_{\textit{finite},m})$, that follows linear sequence dynamics with the p-norm, and there exist L-Lipschitz continuous loss functions $\{f_t : \mathcal{H} \to \mathbb{R}\}_{t=1}^T$ such that the regret of any algorithm $\mathscr{A}$ satisfies*

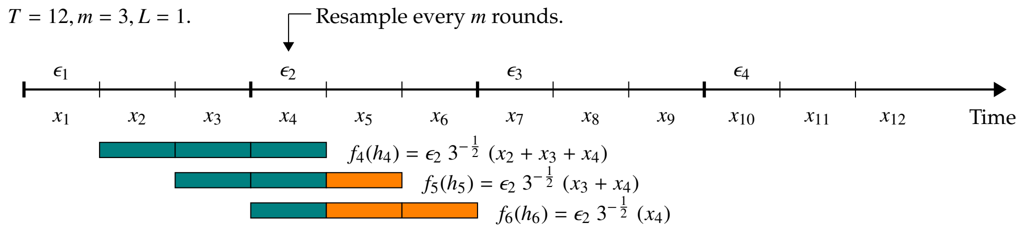
$$\mathsf{Reg}_T(\mathscr{A}) \geq \Omega\left(\sqrt{TL^2 m^{\frac{p+2}{p}}}\right).$$

Proof. Let $\mathcal{X} = [-1, 1]$ and consider an OCO with finite memory problem with constant memory length m, $(\mathcal{X}, \mathcal{H} = \mathcal{X}^m, A_{\text{finite},m}, B_{\text{finite},m})$, that follows linear sequence dynamics with the p-norm. For simplicity, assume that T is a multiple of m (otherwise, the same proof works but with slightly more tedious bookkeeping) and that $L = 1$ (otherwise, multiply the functions f_t defined below by L).

Divide the T rounds into $N = \frac{T}{m}$ blocks of m rounds each. Sample N independent Rademacher random variables $\{\epsilon_1, \ldots, \epsilon_N\}$, where each ϵ_i is equal to ± 1 with probability $\frac{1}{2}$. Recall that $h_t = (x_t, \ldots, x_{t-m+1})$. Define the loss functions $\{f_t\}_{t=1}^T$ as follows. (See Fig. 4.5 for an illustration.) If $t \leq m$, let $f_t = 0$. Otherwise, let

$$\begin{aligned} f_t(h_t) &= \epsilon_{\lceil \frac{t}{m} \rceil} m^{\frac{1-p}{p}} \sum_{k=0}^{m-1-(t-m\lfloor \frac{t}{m} \rfloor - 1)} x_{m\lfloor \frac{t}{m} \rfloor + 1 - k} \\ &= \epsilon_{\lceil \frac{t}{m} \rceil} m^{\frac{1-p}{p}} \left(x_{t-m+1} + \cdots + x_{m\lfloor \frac{t}{m} \rfloor + 1}\right). \end{aligned}$$

In words, the loss in the first m rounds is equal to 0. Thereafter, in round t the loss is equal to a random sign $\epsilon_{\lceil \frac{t}{m} \rceil}$, which is *fixed for that block*, times a scaling factor, which is chosen according to the p-norm to ensure that the Lipschitz constant L is at most 1, times a sum of a *subset* of past decisions in the history $h_t = (x_t, \ldots, x_{t-m+1})$.



$T = 12, m = 3, L = 1.$
Resample every m rounds.
ϵ_1
ϵ_2
ϵ_3
ϵ_4
x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8 x_9 x_{10} x_{11} x_{12}
Time
$f_4(h_4) = \epsilon_2\, 3^{-\frac{1}{2}}\, (x_2 + x_3 + x_4)$
$f_5(h_5) = \epsilon_2\, 3^{-\frac{1}{2}}\, (x_3 + x_4)$
$f_6(h_6) = \epsilon_2\, 3^{-\frac{1}{2}}\, (x_4)$

Figure 4.5: An illustration of the loss functions f_t for the OCO with finite memory lower bound. Suppose $T = 12, m = 3, L = 1$, and $p = 2$. Time is divided into blocks of size $m = 3$. Consider round $t = 5$. The history is $h_5 = (x_3, x_4, x_5)$. The loss function $f_5(h_5)$ is a product of three terms: a random sign ϵ_2 sampled for the block that round 5 belongs to, namely, block 2; a scaling factor of $m^{-\frac{1}{2}}$; a sum over the decisions in the history excluding those that were chosen after observing ϵ_2, i.e., a sum over x_3 and x_4, excluding x_5. The teal and orange shading indicate whether or not a decision was used to construct the loss function in a round. For example, x_3 and x_4 are shaded teal because they are used to construct the loss function f_5. However, x_5 is shaded orange because it is not used to construct f_5.

This subset consists of all past decisions until and including the first decision of the current block, which is the decision in round $m\lfloor \frac{t}{m} \rfloor + 1$.

The functions f_t are linear, so they are convex. In order to show that they satisfy Assumptions **A3** and **A4**, it remains to show that they are 1-Lipschitz continuous. Let $h = (x^{(1)}, \ldots, x^{(m)})$ and $\tilde{h} = (\tilde{x}^{(1)}, \ldots, \tilde{x}^{(m)})$ be arbitrary elements of $\mathcal{H} = \mathcal{X}^m$. We have

$$\begin{aligned}
&\left|f_t(h) - f_t(\tilde{h})\right| \\
&\le \left|\epsilon_{\lceil \frac{t}{m} \rceil} m^{\frac{1-p}{p}} \left((x^{(1)} - \tilde{x}^{(1)}) + \cdots + (x^{(m)} - \tilde{x}^{(m)})\right)\right| \\
&\le m^{\frac{1-p}{p}} \left|(x^{(1)} - \tilde{x}^{(1)}) + \cdots + (x^{(m)} - \tilde{x}^{(m)})\right| && \text{because } \epsilon_{\lceil \frac{t}{m} \rceil} \in \{-1, +1\} \\
&\le m^{\frac{1-p}{p}} m^{1-\frac{1}{p}} \left(\sum_{k=1}^{m} \left|x^{(k)} - \tilde{x}^{(k)}\right|^p\right)^{\frac{1}{p}} && \text{by Hölder's inequality} \\
&= \|h - \tilde{h}\|_{\mathcal{H}},
\end{aligned}$$

where the last equality follows because of our assumption that the problem that follows linear sequence dynamics with the p-norm.

First we will show that the total expected loss of any algorithm is 0, where the expectation is with respect to the randomness in the choice of $\{\epsilon_1, \ldots, \epsilon_N\}$. The total loss in the first block is 0 because $f_t = 0$ for $t \in [m]$. For each subsequent block $n \in \{2, \ldots, N\}$, the total loss in block n depends on the algorithm's choices made *before* observing ϵ_n, namely, $\{x_{(n-2)m+2}, \ldots, x_{(n-1)m+1}\}$. Since ϵ_n is equal to ± 1 with probability $\frac{1}{2}$, the expected loss of any algorithm in a block is equal to 0 and the total expected loss is also equal to 0.

Now we will show that the expected loss of the benchmark is at most

$$-O\left(\sqrt{Tm^{\frac{p+2}{p}}}\right),$$

where the expectation is with respect to the randomness in the choice of

$\{\epsilon_1, \ldots, \epsilon_N\}$. We have

$$\mathbb{E}\left[\min_{x\in\mathcal{X}} \sum_{t=1}^{T} \tilde{f}_t(x)\right] = \mathbb{E}\left[\min_{x\in\mathcal{X}} \sum_{n=2}^{N} \sum_{t=(n-1)m+1}^{nm} \tilde{f}_t(x)\right]$$
$$= \mathbb{E}\left[\min_{x\in\mathcal{X}} \sum_{n=2}^{N} \sum_{t=(n-1)m+1}^{nm} \epsilon_n m^{\frac{1-p}{p}} \times x \times (m-(t-(n-1)m-1))\right].$$

The first equality follows from first summing over blocks and then summing over the rounds in that block. The second equality follows from the definitions of f_t above and of $\tilde{f}_t$ (Definition 4.2.1). By the defintion of $\tilde{f}_t$, the history h_t consists of m copies of x for $t \geq m$.. By the definition of f_t, which sums over all past decisions until the first round of the current block, we have that within a block the sum first extends over m copies of x (in the first round of the block), then $m-1$ copies of x (in the second round of the block), and so on until the last round of the block. So, we have

$$\mathbb{E}\left[\min_{x\in\mathcal{X}} \sum_{t=1}^{T} \tilde{f}_t(x)\right] = \mathbb{E}\left[\min_{x\in\mathcal{X}} \sum_{n=2}^{N} \sum_{t=(n-1)m+1}^{nm} \epsilon_n m^{\frac{1-p}{p}} \times x \times (m-(t-(n-1)m-1))\right]$$
$$= \mathbb{E}\left[\min_{x\in\mathcal{X}} \sum_{n=2}^{N} \sum_{k=0}^{m-1} \epsilon_n m^{\frac{1-p}{p}} \times x \times (m-k)\right]$$
$$= m^{\frac{1-p}{p}} \frac{m^2+m}{2} \mathbb{E}\left[\min_{x\in\mathcal{X}} \sum_{n=2}^{N} \epsilon_n x\right]$$
$$= m^{\frac{1-p}{p}} \frac{m^2+m}{2} \mathbb{E}\left[\min_{x\in\{-1,1\}} \sum_{n=2}^{N} \epsilon_n x\right]$$
$$= m^{\frac{1-p}{p}} \frac{m^2+m}{2} \mathbb{E}\left[\frac{1}{2}\sum_{n=2}^{N} \epsilon_n(-1+1) - \frac{1}{2}\left|\sum_{n=2}^{N} \epsilon_n(-1-1)\right|\right],$$

where the second-last equality follows because the minima of a linear function over an interval is at one of the endpoints and the last equality follows because $\min\{x, y\} = \frac{1}{2}(x + y) - \frac{1}{2}|x - y|$. Since ϵ_n are Rademacher random variables equal to ± 1 with probability $\frac{1}{2}$, we can simplify the above as

$$\begin{aligned}
\mathbb{E}\left[\min_{x\in\mathcal{X}} \sum_{t=1}^{T} \tilde{f}_t(x)\right] &= m^{\frac{1-p}{p}} \frac{m^2+m}{2}\, \mathbb{E}\left[-\frac{1}{2}\left|\sum_{n=2}^{N} -2\epsilon_n\right|\right] \\
&= m^{\frac{1-p}{p}} \frac{m^2+m}{2}\, \mathbb{E}\left[-\left|\sum_{n=2}^{N} \epsilon_n\right|\right] \\
&= -m^{\frac{1-p}{p}} \frac{m^2+m}{2} \mathbb{E}\left[\left|\sum_{n=2}^{N} \epsilon_n\right|\right] \\
&\leq -m^{\frac{1-p}{p}} \frac{m^2+m}{2} \sqrt{N},
\end{aligned}$$

where the last inequality follows from Khintchine's inequality. Using the definition $N = \frac{T}{m}$, we have

$$\begin{aligned}
\mathbb{E}\left[\min_{x\in\mathcal{X}} \sum_{t=1}^{T} \tilde{f}_t(x)\right] &\leq -m^{\frac{1-p}{p}} \frac{m^2+m}{2} \sqrt{\frac{T}{m}} \\
&= -\frac{1}{2}\sqrt{T}\left(m^{\frac{3}{2}+\frac{1-p}{p}} + m^{\frac{1}{2}+\frac{1-p}{p}}\right) \\
&\leq -O\left(\sqrt{T} m^{\frac{3}{2}+\frac{1-p}{p}}\right) \\
&= -O\left(\sqrt{T} m^{\frac{p+2}{2p}}\right) \\
&= -O\left(\sqrt{T m^{\frac{p+2}{p}}}\right).
\end{aligned}$$

Therefore, we have

$$\mathbb{E}_{\epsilon_1,\ldots,\epsilon_N}[\mathsf{Reg}_T(\mathsf{FTRL})] = \mathbb{E}\left[\sum_{t=1}^{T} f_t(h_t)\right] - \mathbb{E}\left[\min_{x\in\mathcal{X}} \sum_{t=1}^{T} \tilde{f}_t(x)\right] \geq \Omega\left(\sqrt{T m^{\frac{p+2}{p}}}\right).$$

This completes the proof. ■

Now we provide a lower bound for the OCO with ρ-discounted infinite memory problem. We restate and prove Theorem 4.7.

Theorem 4.7. *Let $\rho \in [\frac{1}{2}, 1)$. There exists an instance of the online convex optimization with ρ-discounted infinite memory problem, $(\mathcal{X}, \mathcal{H}, A_{\text{infinite},\rho}, B_{\text{infinite}})$, that follows linear sequence dynamics with the 2-norm and there exist L-Lipschitz continuous loss functions $\{f_t : \mathcal{H} \to \mathbb{R}\}_{t=1}^T$ such that the regret of any algorithm $\mathcal{A}$ satisfies*

$$\mathsf{Reg}_T^{\mathsf{OCO-UM}}(\mathcal{A}) \geq \Omega\left(\sqrt{TL^2(1-\rho)^{-2}}\right).$$

The proof is very similar to that of Theorem 4.13 with slight adjustments to account for a ρ-discounted infinite memory instead of a finite memory of constant size m.

Proof. Let $\mathcal{X} = [-1, 1]$ and consider an OCO with infinite memory problem with discount factor ρ, $(\mathcal{X}, \mathcal{H}, A_{\text{infinite},\rho}, B_{\text{infinite}})$, that follows linear sequence dynamics with the 2-norm. For simplicity, assume that T is a multiple of $(1-\rho)^{-1}$ (otherwise, the same proof works but with slightly more tedious bookkeeping) and that $L = 1$ (otherwise, multiply the functions f_t defined below by L).

Define $m = (1-\rho)^{-1}$. Divide the T rounds into $N = \frac{T}{m}$ blocks of m rounds each. Sample N independent Rademacher random variables $\{\epsilon_1, \ldots, \epsilon_N\}$, where each ϵ_i is equal to ± 1 with probability $\frac{1}{2}$. Recall that $h_t = (x_t, \rho x_{t-1}, \ldots, \rho^{t-1} x_1, 0, \ldots)$. Define the loss functions $\{f_t\}_{t=1}^T$ as follows. If $t \leq m$, let $f_t = 0$. Otherwise, let

$$f_t(h_t) = \epsilon_{\lceil \frac{t}{m} \rceil} m^{-\frac{1}{2}} \sum_{k=0}^{m-1} \rho^{k+t-m\lfloor \frac{t}{m} \rfloor - 1} x_{m\lfloor \frac{t}{m} \rfloor + 1 - k}.$$

The functions f_t are linear, so they are convex. In order to show that they satisfy Assumptions **A3** and **A4**, it remains to show that they are 1-Lipschitz continuous. Let $h = (x^{(1)}, \rho x^{(2)}, \dots)$ and $\tilde{h} = (\tilde{x}^{(1)}, \rho \tilde{x}^{(2)}, \dots)$ be arbitrary elements of $\mathcal{H}$. We have

$$\begin{aligned}
&\left| f_t(h) - f_t(\tilde{h}) \right| \\
&\leq \left| \epsilon_{\lceil \frac{t}{m} \rceil} m^{-\frac{1}{2}} \sum_{k=1}^{m} \rho^{k-1} \left(x^{(k)} - \tilde{x}^{(k)} \right) \right| \\
&\leq m^{-\frac{1}{2}} \left| \sum_{k=1}^{m} \rho^{k-1} \left(x^{(k)} - \tilde{x}^{(k)} \right) \right| && \text{because } \epsilon_{\lceil \frac{t}{m} \rceil} \in \{-1, +1\} \\
&\leq m^{-\frac{1}{2}} m^{\frac{1}{2}} \left(\sum_{k=1}^{m} \rho^{2(k-1)} \left| x^{(k)} - \tilde{x}^{(k)} \right|^2 \right)^{\frac{1}{2}} && \text{by Hölder's inequality} \\
&\leq \|h - \tilde{h}\|_{\mathcal{H}},
\end{aligned}$$

where the last equality follows because the follows linear sequence dynamics with the 2-norm.

First we will show that the total expected loss of any algorithm is 0, where the expectation is with respect to the randomness in the choice of $\{\epsilon_1, \dots, \epsilon_N\}$. The total loss in the first block is 0 because $f_t = 0$ for $t \in [m]$. For each subsequent block $n \in \{2, \dots, N\}$, the total loss in block n depends on the algorithm's choices made *before* observing ϵ_n, namely, $\{x_{(n-2)m+2}, \dots, x_{(n-1)m+1}\}$. Since ϵ_n is equal to ± 1 with probability $\frac{1}{2}$, the expected loss of any algorithm in a block is equal to 0 and the total expected loss is also equal to 0.

Now we will show that the expected loss of the benchmark is at most

$$-O\left(\sqrt{T(1-\rho)^{-2}} \right),$$

where the expectation is with respect to the randomness in the choice of $\{\epsilon_1, \ldots, \epsilon_N\}$. We have

$$\begin{aligned}
\mathbb{E}\left[\min_{x\in\mathcal{X}} \sum_{t=1}^{T} \tilde{f}_t(x)\right] &= \mathbb{E}\left[\min_{x\in\mathcal{X}} \sum_{n=2}^{N} \sum_{t=(n-1)m+1}^{nm} \tilde{f}_t(x)\right] \\
&= \mathbb{E}\left[\min_{x\in\mathcal{X}} \sum_{n=2}^{N} \sum_{t=(n-1)m+1}^{nm} \epsilon_n m^{-\frac{1}{2}} \sum_{k=0}^{m-1} \rho^{k+t-(n-1)m-1} x\right] \\
&= m^{-\frac{1}{2}} \mathbb{E}\left[\min_{x\in\mathcal{X}} \sum_{n=2}^{N} \epsilon_n x \sum_{t=(n-1)m+1}^{nm} \rho^{t-(n-1)m-1} \sum_{k=0}^{m-1} \rho^k\right] \\
&= m^{-\frac{1}{2}} \frac{1-\rho^m}{1-\rho} \mathbb{E}\left[\min_{x\in\mathcal{X}} \sum_{n=2}^{N} \epsilon_n x \sum_{t=(n-1)m+1}^{nm} \rho^{t-(n-1)m-1}\right] \\
&= m^{-\frac{1}{2}} \left(\frac{1-\rho^m}{1-\rho}\right)^2 \underbrace{\mathbb{E}\left[\min_{x\in\mathcal{X}} \sum_{n=2}^{N} \epsilon_n x\right]}_{(a)}.
\end{aligned}$$

The term (a) above can be bounded above by $-\sqrt{N}$ as in the proof of Theorem 4.13 using Khintchine's inequality. Therefore, using that $N = \frac{T}{m}$ and $m = (1-\rho)^{-1}$ we have

$$\begin{aligned}
\mathbb{E}\left[\min_{x\in\mathcal{X}} \sum_{t=1}^{T} \tilde{f}_t(x)\right] &\leq -m^{-\frac{1}{2}} \left(\frac{1-\rho^m}{1-\rho}\right)^2 \sqrt{N} \\
&\leq -(1-\rho)^{\frac{1}{2}} \left(\frac{1-\rho^m}{1-\rho}\right)^2 \sqrt{T(1-\rho)} \\
&= -\sqrt{T}\frac{(1-\rho^m)^2}{1-\rho} \\
&= -\sqrt{T(1-\rho)^{-2}}(1-\rho^m)^2 \\
&\leq -O\left(\sqrt{T(1-\rho)^{-2}}\right),
\end{aligned}$$

where the last inequality follows from the assumption that $\rho \in [\frac{1}{2}, 1)$ and the following argument:

$$\begin{aligned}
\rho^m = (1-(1-\rho))^m = (1-(1-\rho))^{\frac{1}{1-\rho}} \leq \frac{1}{e} \\
\Rightarrow (1-\rho^m) \geq 1 - \frac{1}{e} \\
\Rightarrow (1-\rho^m)^2 \geq \left(1-\frac{1}{e}\right)^2 \\
\Rightarrow -(1-\rho^m)^2 \leq -\left(1-\frac{1}{e}\right)^2
\end{aligned}$$

This completes the proof. ■

4.12 Proofs for Section 4.4: Online Linear Control

We use the following standard facts about matrix norms.

Lemma 4.12.1. *Let $M, N \in \mathbb{R}^{d\times d}$. Then,*

1. $\|M\|_2 \leq \|M\|_F \leq \sqrt{d}\|M\|_2$.
2. $\|MN\|_F \leq \|M\|_2\|N\|_F$.

Proof. Part 1 can be found in, for example, Golub and Loan [1996, Section 2.3.2]. Letting N_j denote the j-th column of N, part 2 follows from

$$\|MN\|_F^2 = \sum_{j=1}^{d} \|MN_j\|_2^2 \leq \|M\|_2^2 \sum_{j=1}^{d} \|N_j\|_2^2 = \|M\|_2^2\|N\|_F^2.$$

This completes the proof. ■

Lemma 4.12.2. *For $j \geq 2$, the operator norm $\|A^j\|$ is bounded above as*

$$\left\|A^j\right\| \leq O\left(\kappa^4 \rho^{\frac{j}{2}}\right).$$

Proof. Recall the definition of $\mathcal{H}$ and $\|\cdot\|_{\mathcal{H}}$ (Eq. (4.4)). Let

$$(Y_0, Y_1, \dots) = (Y_0, GX_1, \widetilde{F}GX_2, \widetilde{F}^2GX_3, \dots)$$

be an element of $\mathcal{H}$ with unit norm, i.e.,

$$\sqrt{\sum_{k=0}^{\infty} \xi_k^2 \|Y_k\|_{\mathcal{X}}^2} = 1,$$

where the weights (ξ_k) are defined in Eq. (4.5). Note that $\xi_p = 1$ for $p = 0, 1$ and $\xi_p^2 = \rho^{-p+2}$ for $p = 2, 3, \dots$. From the definition of the operator A and for $j \geq 2$, we have

$$A^j((Y_0, Y_1, \dots)) = (0, \dots, 0, \widetilde{F}^{j-1}GY_0, \widetilde{F}^jGX_1, \widetilde{F}^{j+1}GX_2, \dots).$$

Now we bound $\|A^j\|$ as follows. By definition of A^j and $\|\cdot\|_{\mathcal{H}}$ (Eq. (4.4)), and part 2 of Lemma 4.12.1, we have

$$\begin{aligned}
\left\|A^j((Y_0, Y_1, \dots))\right\| &= \sqrt{\rho^{-j+2}\|\widetilde{F}^{j-1}GY_0\|_{\mathcal{X}}^2 + \sum_{k=1}^{\infty} \rho^{-j-k+2}\|\widetilde{F}^{j+k-1}GX_k\|_{\mathcal{X}}^2} \\
&\leq \sqrt{\rho^{-j+2}\|\widetilde{F}^{j-1}G\|_2^2\|Y_0\|_{\mathcal{X}}^2 + \sum_{k=1}^{\infty} \rho^{-j-k+2}\|\widetilde{F}^{j-1}\|_2^2\|\widetilde{F}\|_2^2\|\widetilde{F}^{k-1}GX_k\|_{\mathcal{X}}^2} \\
&\leq \rho^{-\frac{j}{2}}\|\widetilde{F}^{j-1}\|_2 \sqrt{\rho^2\|G\|_2^2\|Y_0\|_{\mathcal{X}}^2 + \sum_{k=1}^{\infty} \rho^{-k+2}\|\widetilde{F}\|_2^2\|\widetilde{F}^{k-1}GX_k\|_{\mathcal{X}}^2} \\
&= \rho^{-\frac{j}{2}}\|\widetilde{F}^{j-1}\|_2 \sqrt{\rho^2\|G\|_2^2\|Y_0\|_{\mathcal{X}}^2 + \sum_{k=1}^{\infty} \rho^{-k+2}\|\widetilde{F}\|_2^2\|Y_k\|_{\mathcal{X}}^2}.
\end{aligned}$$

Using our assumptions that $\|G\|_2 \leq \kappa$ and $\|\widetilde{F}\|_2 \leq \kappa^2\rho$, we have

$$\begin{aligned}
\left\|A^j((Y_0, Y_1, \dots))\right\| &\leq \rho^{-\frac{j}{2}} \|\widetilde{F}^{j-1}\|_2 \sqrt{\rho^2\kappa^2\|Y_0\|_{\mathcal{X}}^2 + \sum_{k=1}^{\infty} \rho^{-k+2}\kappa^4\rho^2\|Y_k\|_{\mathcal{X}}^2} \\
&\leq \rho^{-\frac{j}{2}} \rho\kappa^2 \|\widetilde{F}^{j-1}\|_2 \sqrt{\|Y_0\|_{\mathcal{X}}^2 + \sum_{k=1}^{\infty} \rho^{-k+2}\|Y_k\|_{\mathcal{X}}^2} \\
&\leq \rho^{-\frac{j}{2}} \rho\kappa^2\kappa^2\rho^{j-1} \sqrt{\|Y_0\|_{\mathcal{X}}^2 + \sum_{k=1}^{\infty} \rho^{-k+2}\|Y_k\|_{\mathcal{X}}^2} \\
&= \kappa^4\rho^{\frac{j}{2}} \sqrt{\|Y_0\|_{\mathcal{X}}^2 + \sum_{k=1}^{\infty} \rho^{-k+2}\|Y_k\|_{\mathcal{X}}^2}.
\end{aligned}$$

Using $\rho^{-1+2} = \rho < 1$ for $k = 1$ in the above sum, the definition of (ξ_k), and our assumption that $(Y_0, Y_1 \dots)$ has unit norm, we have

$$\left\|A^j((Y_0, Y_1, \dots))\right\| \leq \kappa^4\rho^{\frac{j}{2}} \sqrt{\xi_0^2\|Y_0\|_{\mathcal{X}}^2 + \sum_{k=1}^{\infty} \xi_k^2\|Y_k\|_{\mathcal{X}}^2} = \kappa^4\rho^{\frac{j}{2}}.$$

This completes the proof. ■

Lemma 4.12.3. *The* 2*-effective memory capacity is bounded above as*

$$H_2 \leq O\left(\kappa^4(1-\rho)^{-\frac{3}{2}}\right).$$

Proof. Using Lemma 4.12.2 to bound $\|A^k\|$ for $k \geq 2$, we have

$$H_2 = \sqrt{\sum_{k=0}^{\infty} k^2\|A^k\|^2} \leq O\left(\sqrt{\sum_{k=2}^{\infty} k^2\kappa^8\rho^k}\right) \leq O\left(\kappa^4(1-\rho)^{-\frac{3}{2}}\right).$$ ■

Lemma 4.12.4. *Suppose $R : \mathcal{X} \to \mathbb{R}$ is defined as $R(M) = \frac{1}{2}\|M\|_{\mathcal{X}}^2$. Then, it is* 1*-strongly-convex and $D = \max_{M,\widetilde{M}\in\mathcal{X}} |R(M) - R(\widetilde{M})| \leq d\kappa^8(1-\rho)^{-1}$.*

Proof. Note that R is 1-strongly-convex by definition. Using part 1 of Lemma 4.12.1 and the definition of $\mathcal{X}$ (Eq. (4.2)), we have for all $M, \widetilde{M} \in \mathcal{X}$,

$$\begin{aligned}
D &= \max_{M,\widetilde{M}\in\mathcal{X}} |R(M) - R(\widetilde{M})| \\
&= \max_{M,\widetilde{M}\in\mathcal{X}} \left|\frac{1}{2}\|M\|_{\mathcal{X}}^2 - \frac{1}{2}\|\widetilde{M}\|_{\mathcal{X}}^2\right| \\
&\le \max_{M\in\mathcal{X}} \|M\|_{\mathcal{X}}^2 \\
&= \max_{M\in\mathcal{X}} \sum_{j=1}^{\infty} \rho^{-j}\|M^{[j]}\|_F^2 && \text{by Eq. (4.3)} \\
&\le \max_{M\in\mathcal{X}} \sum_{j=1}^{\infty} \rho^{-j} d\|M^{[j]}\|_2^2 && \text{by Lemma 4.12.1} \\
&\le \sum_{j=1}^{\infty} \rho^{-j} d\kappa^8\rho^{2j} && \text{by Eq. (4.2)} \\
&\le d\kappa^8(1-\rho)^{-1}.
\end{aligned}$$

This completes the proof. ∎

Lemma 4.12.5. *We can bound the norm of the state and control at time t as*

$$\max\{\|s_t\|_2, \|u_t\|_2\} \le D_{\mathcal{X}} = O\left(W\kappa^8(1-\rho)^{-2}\right).$$

Proof. We can bound the norm of s_t and u_t using Eqs. (4.6) and (4.7) as

$$\begin{aligned}
\|s_t\|_2 &\le \left\| \widetilde{F}^{t-1} s_1 + \sum_{k=1}^{t-1} \sum_{j=1}^{k} \widetilde{F}^{t-k-1} G M_k^{[j]} w_{k-j} + w_{t-1} \right\|_2 \\
&\le \kappa^2 \rho^{t-1} + W + \sum_{k=1}^{t} \sum_{j=1}^{k} \kappa^2 \rho^{t-k-1} \kappa \kappa^4 \rho^j W \\
&\le \kappa^2 + W + W\kappa^7 (1-\rho)^{-2} \\
&\le O\left(W\kappa^7 (1-\rho)^{-2}\right). \\
\|u_t\|_2 &\le \left\| K s_t + \sum_{j=1}^{t} M_t^{[j]} w_{t-j} \right\|_2 \\
&\le O\left(W\kappa^8 (1-\rho)^{-2}\right) + \sum_{j=1}^{t} W\kappa^4 \rho^j \\
&\le O\left(W\kappa^8 (1-\rho)^{-2}\right).
\end{aligned}$$

Above, we used the assumptions that $\rho < 1$ and $\kappa, W \ge 1$. This completes the proof. ■

Lemma 4.12.6. *The Lipschitz constant of f_t can be bounded above as*

$$L \le O\left(L_0 D_{\mathcal{X}} W \kappa (1-\rho)^{-1}\right),$$

where $D_{\mathcal{X}}$ is defined in Lemma 4.12.5.

Proof. Let $(M_1, \ldots, M_t)$ and $(\widetilde{M}_1, \ldots, \widetilde{M}_t)$ be two sequences of decisions, where M_k and $\widetilde{M}_k \in \mathcal{X}$. Let h_t and $\tilde{h}_t$ be the corresponding histories, and (s_t, u_t) and $(\tilde{s}_t, \tilde{u}_t)$ be the corresponding state-control pairs at the end of round t. We have

$$\begin{aligned}\left|f_t(h_t) - f_t(\tilde{h}_t)\right| &= |c_t(s_t, u_t) - c_t(\tilde{s}_t, \tilde{u}_t)| \\ &\leq L_0 D_{\mathcal{X}} \max\{\|s_t - \tilde{s}_t\|_2\,, \|u_t - \tilde{u}_t\|_2\},\end{aligned}$$

where the last inequality follows from our assumptions about the functions c_t and Lemma 4.12.5. It suffices to bound the two norms on the right-hand side in terms of $\|h_t - \tilde{h}_t\|_{\mathcal{H}}$. For $k \in \{1, \ldots, t-1\}$, define $Z_k^{[s]} = \widetilde{F}^{t-k-1} G(M_k^{[s]} - \widetilde{M}_k^{[s]})$. Using Eq. (4.6), we have

$$\begin{aligned}\|s_t - \tilde{s}_t\|_2 &= \left\|\sum_{k=1}^{t-1}\sum_{j=1}^{k} Z_k^{[j]} w_{k-j}\right\|_2 \\ &\leq \sum_{k=1}^{t-1}\sum_{j=1}^{k} \left\|Z_k^{[j]} w_{k-j}\right\|_2 \\ &= \sum_{k=1}^{t-1}\sum_{j=1}^{k} \left\|\rho^{-\frac{j}{2}} Z_k^{[j]} \rho^{\frac{j}{2}} w_{k-j}\right\|_2 \\ &\leq \sum_{k=1}^{t-1}\sum_{j=1}^{k} \left\|\rho^{-\frac{j}{2}} Z_k^{[j]}\right\|_2 \left\|\rho^{\frac{j}{2}} w_{k-j}\right\|_2 \\ &= \sum_{k=1}^{t-1}\sum_{j=1}^{k} \xi_{1+t-1-k} \left\|\rho^{-\frac{j}{2}} Z_k^{[j]}\right\|_2 \xi_{1+t-1-k}^{-1} \left\|\rho^{\frac{j}{2}} w_{k-j}\right\|_2 \\ &\leq \sqrt{\sum_{k=1}^{t-1}\sum_{j=1}^{k} \xi_{t-k}^2 \left\|\rho^{-\frac{j}{2}} Z_k^{[j]}\right\|_2^2}\sqrt{\sum_{k=1}^{t-1}\sum_{j=1}^{k} \xi_{t-k}^{-2} \left\|\rho^{\frac{j}{2}} w_{k-j}\right\|_2^2} \qquad (4.11) \\ &= \underbrace{\sqrt{\sum_{k=1}^{t-1} \xi_{t-k}^2 \sum_{j=1}^{k} \left\|\rho^{-\frac{j}{2}} Z_k^{[j]}\right\|_2^2}}_{(a)}\underbrace{\sqrt{\sum_{k=1}^{t-1} \xi_{t-k}^{-2} \sum_{j=1}^{k} \left\|\rho^{\frac{j}{2}} w_{k-j}\right\|_2^2}}_{(b)},\end{aligned}$$

where Eq. (4.11) follows from the Cauchy-Schwarz inequality. The specific choice of weighted norms on $\mathcal{X}$ and $\mathcal{H}$ allow us to bound the terms (a) and (b) in terms of $\|h_t - \tilde{h}_t\|_{\mathcal{H}}$. We can bound the term (a) using the definition of $Z_k^{[s]}$, $\|\cdot\|_{\mathcal{X}}$, and $\|\cdot\|_{\mathcal{H}}$ as

$$\begin{aligned}\sqrt{\sum_{k=1}^{t-1}\xi_{t-k}^2\sum_{j=1}^{k}\left\|\rho^{-\frac{j}{2}}Z_k^{[j]}\right\|_2^2} &= \sqrt{\sum_{k=1}^{t-1}\xi_{t-k}^2\sum_{j=1}^{k}\rho^{-j}\left\|\widetilde{F}^{t-k-1}G(M_k^{[j]}-\widetilde{M}_k^{[j]})\right\|_2^2}\\ &\leq \sqrt{\sum_{k=1}^{t-1}\xi_{t-k}^2\sum_{j=1}^{k}\rho^{-j}\left\|\widetilde{F}^{t-k-1}G(M_k^{[j]}-\widetilde{M}_k^{[j]})\right\|_F^2} \qquad (4.12)\\ &\leq \|h_t-\tilde{h}_t\|_{\mathcal{H}}, \qquad (4.13)\end{aligned}$$

where Eq. (4.12) follows from part 1 of Lemma 4.12.1 and Eq. (4.13) follows from the definitions of $\|\cdot\|_{\mathcal{X}}$ and $\|\cdot\|_{\mathcal{H}}$. Using $\|w_t\|_2 \leq W$ for all rounds t, we can bound the term (b) as

$$\begin{aligned}\sqrt{\sum_{k=1}^{t-1}\xi_{t-k}^{-2}\sum_{j=1}^{k}\left\|\rho^{\frac{j}{2}}w_{k-j}\right\|_2^2} &\leq W\sqrt{\sum_{k=1}^{t-1}\xi_{t-k}^{-2}\sum_{j=1}^{k}\rho^{j}}\\ &\leq W\sqrt{\sum_{k=1}^{t-1}\xi_{t-k}^{-2}\frac{\rho(1-\rho^{k+1})}{1-\rho}}\\ &\leq W(1-\rho)^{-1}, \qquad (4.14)\end{aligned}$$

where Eq. (4.14) follows from the definition of (ξ_k) (Eq. (4.5)). Substituting Eqs. (4.13) and (4.14) in Eq. (4.11), we have

$$\|s_t - \tilde{s}_t\|_2 \leq W(1-\rho)^{-1}\|h_t - \tilde{h}_t\|_{\mathcal{H}}.$$

Similarly,

$$\begin{aligned}\|u_t - \tilde{u}_t\| &= \left\|K(s_t - \tilde{s}_t) + \sum_{j=1}^{t}(M_t^{[j]} - \widetilde{M}_t^{[j]})w_{t-j}\right\|_2\\ &\leq O\left(W\kappa(1-\rho)^{-1}\left\|h_t - \tilde{h}_t\right\|_{\mathcal{H}}\right),\end{aligned}$$

where the last inequality follows from our assumption that $\|K\|_2 \leq \kappa$ and the above inequality for $\|s_t - \tilde{s}_t\|_2$. This completes the proof. ■

Lemma 4.12.7. *The Lipschitz constant of $\tilde{f}_t$ can be bounded above as*

$$\tilde{L} \leq O\left(L_0 D_{\mathcal{X}} W \kappa^5 (1-\rho)^{-\frac{3}{2}}\right),$$

where $D_{\mathcal{X}}$ is defined in Lemma 4.12.5.

Proof. Using Lemma 4.12.2 that bounds $\|A^k\|$, we have

$$\sqrt{\sum_{k=0}^{\infty} \left\|A^k\right\|^2} \leq O\left(\kappa^4 (1-\rho)^{-\frac{1}{2}}\right).$$

Using Theorem 4.1 that bounds $\tilde{L}$ in terms of L and the above, we have

$$\tilde{L} \leq O\left(L\kappa^4 (1-\rho)^{-\frac{1}{2}}\right) \leq O\left(L_0 D_{\mathcal{X}} W \kappa^5 (1-\rho)^{-\frac{3}{2}}\right),$$

where the last inequality follows from Lemma 4.12.6. ■

Now we restate and prove Theorem 4.8.

Theorem 4.8. *Consider the online linear control problem as defined in Section 4.4.1. Suppose the decisions in round t are chosen using Algorithm 4. Then, the upper bound on the policy regret is*

$$\mathsf{Reg}_T^{\mathsf{OLC}}(\mathsf{FTRL}) = O\left(L_0 W^2 \sqrt{T} d^{\frac{1}{2}} \kappa^{17} (1-\rho)^{-4.5}\right). \tag{4.8}$$

Proof. Using Theorem 4.2 and the above lemmas, we can upper bound the policy regret of Algorithm 4 for the online linear control problem by

$$
\begin{aligned}
&O\left(\sqrt{\frac{D}{\alpha} T L \tilde{L} H_2}\right) \\
&= O\left(\sqrt{d\kappa^8(1-\rho)^{-1}\ T\ \left(L_0 W^2 \kappa^9 (1-\rho)^{-3}\right)^2 \kappa^4 (1-\rho)^{-\frac{1}{2}}\ \kappa^4(1-\rho)^{-\frac{3}{2}}}\right) \\
&= O\left(L_0 W^2 \sqrt{T} d^{\frac{1}{2}} \kappa^{17} (1-\rho)^{-4.5}\right).
\end{aligned}
$$

This completes the proof. ■

Existing Regret Bound. The upper bound on policy regret for the online linear control problem in existing work is given in Agarwal et al. [2019b, Theorem 5.1]. The theorem statement only shows the dependence on $\tilde{L}, W$, and T. The dependence on d, κ, and ρ can be found in the details of the proof. Below we give a detailed accounting of all of these terms in their regret bound.

To simplify notation let $\gamma = 1-\rho$. Agarwal et al. [2019b] define

$$
H = \frac{\kappa^2}{\gamma}\log(T) \quad \text{and} \quad C = \frac{W(\kappa^2 + H\kappa_B \kappa^2 a)}{\gamma(1-\kappa^2(1-\gamma)^{H+1})} + \frac{\kappa_B \kappa^3 W}{\gamma}.
$$

The value of a is not specified in Theorem 5.1. However, from Theorem 5.3 and the definition of $\mathcal{M}$ in Algorithm 1 their paper, we can infer that $a = \kappa_B \kappa^3$.

The final regret bound is obtained by summing Equations 5.1, 5.3, and 5.4. Given the definition of H above, we have that

$$
(1-\gamma)^{H+1} \leq \exp(-\kappa^2 \log T) = T^{-\kappa^2}.
$$

So, the dominant term in the regret bound is Equation 5.4, which is

$$O\left(L_0 W C d^{\frac{3}{2}} \kappa_B^2 \kappa^6 H^{2.5} \gamma^{-1} \sqrt{T}\right).$$

Substituting the values of H and C from above and collecting terms, we have that the upper bound on policy regret in existing work [Agarwal et al., 2019b, Theorem 5.1] is

$$\begin{aligned}
&O\left(L_0 W d^{\frac{3}{2}} \sqrt{T} \log(T)^{2.5} \kappa_B^2 \kappa^{11} \gamma^{-3.5} C\right) \\
&= O\left(L_0 W d^{\frac{3}{2}} \sqrt{T} \log(T)^{2.5} \kappa_B^2 \kappa^{11} \gamma^{-3.5} \left(\frac{W(\kappa^2 + H\kappa_B \kappa^2 a)}{\gamma(1-\kappa^2(1-\gamma)^{H+1})} + \frac{\kappa_B \kappa^3 W}{\gamma}\right)\right) \\
&= O\left(L_0 W d^{\frac{3}{2}} \sqrt{T} \log(T)^{2.5} \kappa_B^2 \kappa^{11} \gamma^{-3.5} \left(\frac{W\kappa^2}{\gamma(1-\kappa^2(1-\gamma)^{H+1})} + \frac{W\kappa_B^2 \kappa^7 \log(T)}{\gamma^2(1-\kappa^2(1-\gamma)^{H+1})} + \frac{\kappa_B \kappa^3 W}{\gamma}\right)\right) \\
&= O\left(L_0 W^2 d^{\frac{3}{2}} \sqrt{T} \log(T)^{2.5} \kappa_B^2 \kappa^{13} \gamma^{-4.5} (1-\kappa^2(1-\gamma)^{H+1})^{-1}\right) \\
&\quad + O\left(L_0 W^2 d^{\frac{3}{2}} \sqrt{T} \log(T)^{3.5} \kappa_B^4 \kappa^{18} \gamma^{-5.5} (1-\kappa^2(1-\gamma)^{H+1})^{-1}\right) \\
&\quad + O\left(L_0 W^2 d^{\frac{3}{2}} \sqrt{T} \log(T)^{2.5} \kappa_B^3 \kappa^{14} \gamma^{-4.5}\right) \\
&= O\left(L_0 W^2 d^{\frac{3}{2}} \sqrt{T} \log(T)^{3.5} \kappa_B^4 \kappa^{18} \gamma^{-5.5}\right).
\end{aligned}$$

Above we used that $\lim_{T\to\infty}(1-\kappa^2(1-\gamma)^{H+1})^{-1} = 1$ to simplify the expressions. Therefore, the upper bound on policy regret for the online linear control problem in existing work is

$$O\left(L_0 W^2 d^{\frac{3}{2}} \sqrt{T} \log(T)^{3.5} \kappa_B^4 \kappa^{18} \gamma^{-5.5}\right). \tag{4.15}$$

Our regret bound in Theorem 4.8 is

$$O\left(L_0 W^2 \sqrt{T} d^{\frac{1}{2}} \kappa^{17} (1-\rho)^{-4.5}\right).$$

Therefore, our bound improves upon existing work by a factor of

$$O\left(\log(T)^{3.5} d\kappa^5 (1-\rho)^{-1}\right),$$

where we note that we assumed $\kappa_B = \kappa$ in our work for simplicity and we defined $\gamma = 1 - \rho$.

4.13 Proofs for Section 4.4: Online Performative Prediction

Before formulating the online performative prediction problem in our OCO with unbounded memory framework, we state the definition of 1-Wasserstein distance that we use in our regret analysis. Informally, the 1-Wasserstein distance is a measure of the distance between two probability measures.

Definition 4.13.1 (1-Wasserstein Distance)**.** Let $(\mathcal{Z}, d)$ be a metric space. Let $\mathbb{P}(\mathcal{Z})$ denote the set of Radon probability measures ν on $\mathcal{Z}$ with finite first moment. That is, there exists $z' \in \mathcal{Z}$ such that $\mathbb{E}_{z\sim\nu}[d(z, z')] < \infty$. The 1-Wasserstein distance between two probability measures $\nu, \nu' \in \mathbb{P}(\mathcal{Z})$ is defined as

$$W_1(\nu, \nu') = \sup\{\mathbb{E}_{z\sim\nu}[f(z)] - \mathbb{E}_{z\sim\nu'}[f(z)]\},$$

where the supremum is taken over all 1-Lipschitz continuous functions $f : \mathcal{Z} \to \mathbb{R}$.

Lemma 4.13.1. *The operator norm $\|A^s\|$ is bounded above as*

$$\|A^s\| \leq O(\rho^s).$$

Proof. Recall the definition of $\mathcal{H}$ and $\|\cdot\|_{\mathcal{H}}$. Let

$$(y_0, y_1, \dots) = (x_0, \rho x_1, \rho^2 x_2, \dots)$$

be an element of $\mathcal{H}$ with unit norm, i.e.,

$$\sum_{k=0}^{\infty} \|y_k\| = 1.$$

From the definition of the operator A, we have

$$A^s((y_0, y_1, \dots)) = (0, \dots, 0, \rho^s x_0, \rho^{s+1} x_1, \dots).$$

Now we bound $\|A^s\|$ as follows. By definition of A^s and $\|\cdot\|_{\mathcal{H}}$, we have

$$\|A^s((y_0, y_1, \dots))\| = \sum_{k=0}^{\infty} \rho^{s+k}\|x_k\| = \rho^s \sum_{k=0}^{\infty} \rho^k \|x_k\| = \rho^s \sum_{k=0}^{\infty} \|y_k\| = \rho^s. \qquad \blacksquare$$

Lemma 4.13.2. *The* 1*-effective memory capacity is bounded above as*

$$H_2 \le O\left((1-\rho)^{-2}\right).$$

Proof. Using Lemma 4.13.1 to bound $\|A^k\|$, we have

$$H_1 = \sum_{k=0}^{\infty} k\|A^k\| = \sum_{k=0}^{\infty} k\rho^k \le O\left((1-\rho)^{-2}\right). \qquad \blacksquare$$

Lemma 4.13.3. *Suppose $R : \mathcal{X} \to \mathbb{R}$ is defined as $R(x) = \frac{1}{2}\|x\|_{\mathcal{X}}^2$. Then, it is* 1*-strongly-convex and $D = \max_{x,\tilde{x} \in \mathcal{X}} |R(x) - R(\tilde{x})| \le D_{\mathcal{X}}^2$.*

Proof. Note that R is 1-strongly-convex by definition. By the assumption that $\|x\|_{\mathcal{X}} \le D_{\mathcal{X}}$ for all $x \in \mathcal{X}$, we have that $D \le D_{\mathcal{X}}^2$. $\blacksquare$

Lemma 4.13.4. *The Lipschitz constant of f_t can be bounded above as*

$$L \leq O\left(L_0 \frac{1-\rho}{\rho} \|F\|_2\right).$$

Proof. Let $(x_1, \ldots, x_t)$ and $(\tilde{x}_1, \ldots, \tilde{x}_t)$ be two sequences of decisions, where $x_k, \tilde{x}_k \in \mathcal{X}$. Let h_t and $\tilde{h}_t$ be the corresponding histories, and p_t and $\tilde{p}_t$ be the corresponding distributions at the end of round t. We have

$$\begin{aligned}
&\left|f_t(h_t) - f_t(\tilde{h}_t)\right| \\
&= \left|\mathbb{E}_{z\sim p_t}\left[l_t(x_t, z)\right] - \mathbb{E}_{z\sim \tilde{p}_t}\left[l_t(\tilde{x}_t, z)\right]\right| \\
&= \left|\mathbb{E}_{z\sim p_t}\left[l_t(x_t, z)\right] - \mathbb{E}_{z\sim p_t}\left[l_t(\tilde{x}_t, z)\right] + \mathbb{E}_{z\sim p_t}\left[l_t(\tilde{x}_t, z)\right] - \mathbb{E}_{z\sim \tilde{p}_t}\left[l_t(\tilde{x}_t, z)\right]\right| \\
&\leq L_0\|x_t - \tilde{x}_t\|_2 + L_0 W_1(p_t, \tilde{p}_t),
\end{aligned}$$

where the last inequality follows from the assumptions about the functions l_t and the definition of the Wasserstein distance W_1. By definition of p_t (Eq. (4.9)), we have

$$\begin{aligned}
W_1(p_t, \tilde{p}_t) &\leq \sum_{k=1}^{t-1} \frac{1-\rho}{\rho} \rho^k \|F\|_2 \|x_{t-k} - \tilde{x}_{t-k}\|_2 \\
&\leq \frac{1-\rho}{\rho} \|F\|_2 \|h_t - \tilde{h}_t\|_{\mathcal{H}},
\end{aligned}$$

where the last inequality follows from the definition of $\|\cdot\|_{\mathcal{H}}$. Therefore, $L \leq L_0 \frac{1-\rho}{\rho} \|F\|_2$. ■

Lemma 4.13.5. *The Lipschitz constant of f_t can be bounded above as*

$$\tilde{L} \leq O\left(L_0 \frac{1}{\rho} \|F\|_2\right).$$

Proof. Using Lemma 4.13.1 that bounds $\|A^k\|$, we have

$$\sum_{k=0}^{\infty} \|A^k\| = (1-\rho)^{-1}.$$

Using Theorem 4.1 that bounds $\tilde{L}$ in terms of L and the above, we have

$$\tilde{L} \leq O\left(L(1-\rho)^{-1}\right) = O\left(L_0 \frac{1}{\rho} \|F\|_2\right),$$

where the last equality follows from Lemma 4.13.4. ■

Now we restate and prove Theorem 4.9.

Theorem 4.9. *Consider the online performative prediction problem as defined in Section 4.4.2. Suppose the decisions in round t are chosen using Algorithm 4. Then, the upper bound on the policy regret is*

$$\mathsf{Reg}_T^{\mathsf{OPP}}(\mathsf{FTRL}) = O\left(D_{\mathcal{X}} L_0 \sqrt{T} \|F\|_2 (1-\rho)^{-\frac{1}{2}} \rho^{-1}\right).$$

Proof. Using Theorem 4.2 and the above lemmas, we can upper bound the policy regret of Algorithm 4 for the online performative prediction problem by

$$O\left(\sqrt{\frac{D}{\alpha} T L \tilde{L} H_1}\right) = O\left(D_{\mathcal{X}} L_0 \|F\|_2 (1-\rho)^{-\frac{1}{2}} \rho^{-1} \sqrt{T}\right).$$

This completes the proof. ■

We note that the upper bound can be improved by defining a weighted norm on $\mathcal{H}$ similar to the approach in Section 4.12. However, here we present the looser anaysis for simplicity of exposition.

4.14 Proofs for Section 4.6: Mini-Batch FTRL

We restate and prove Theorem 4.10.

Theorem 4.10. *Consider an online convex optimization with unbounded memory problem specified by $(\mathcal{X}, \mathcal{H}, A, B)$. Let the regularizer $R : \mathcal{X} \to \mathbb{R}$ be α-strongly-convex and satisfy $|R(x) - R(\tilde{x})| \le D$ for all $x, \tilde{x} \in \mathcal{X}$. Algorithm 5 with batch size S and step-size η satisfies*

$$\mathsf{Reg}_T^{\mathsf{OCO-UM}}(\mathsf{Mini-Batch\ FTRL}) \le \frac{SD}{\eta} + \eta\frac{T\tilde{L}^2}{\alpha} + \eta\frac{TL\tilde{L}H_1}{S\alpha}.$$

If $\eta = \sqrt{\frac{\alpha SD}{T\tilde{L}\left(\frac{LH_1}{S}+\tilde{L}\right)}}$, then

$$\mathsf{Reg}_T^{\mathsf{OCO-UM}}(\mathsf{Mini-Batch\ FTRL}) \le O\left(\sqrt{\frac{D}{\alpha}T\left(L\tilde{L}H_1 + S\tilde{L}^2\right)}\right).$$

Proof. For simplicity, assume that T is a multiple of S. Otherwise, the same proof works after replacing $\frac{T}{S}$ with $\lceil\frac{T}{S}\rceil$. Let $x^* \in \arg\min_{x\in\mathcal{X}} \sum_{t=1}^T \tilde{f}_t(x)$. Note that we can write the regret as

$$\begin{aligned}\mathsf{Reg}_T(\mathsf{Mini-Batch\ FTRL}) &= \sum_{t=1}^T f_t(h_t) - \min_{x\in\mathcal{X}}\sum_{t=1}^T \tilde{f}_t(x) \\ &= \underbrace{\sum_{t=1}^T f_t(h_t) - \tilde{f}_t(x_t)}_{(a)} + \underbrace{\sum_{t=1}^T \tilde{f}_t(x_t) - \tilde{f}_t(x^*)}_{(b)}.\end{aligned}$$

We can bound the term (b) using Theorem 2.1 for mini-batches [Dekel et al., 2012, Altschuler and Talwar, 2018, Chen et al., 2020] by

$$\frac{SD}{\eta} + \eta\frac{T\tilde{L}^2}{\alpha}.$$

It remains to bound term (a). Let $N = {}^{T}/_{S}$ denote the number of batches and $T_n = \{(n-1)S + 1, \dots, nS\}$ denote the rounds in batch $n \in [N]$. We can write

$$\begin{aligned}
\sum_{t=1}^{T} f_t(h_t) - \tilde{f}_t(x_t) &= \sum_{t=1}^{T} f_t\left(\sum_{k=0}^{t-1} A^k B x_{t-k}\right) - f_t\left(\sum_{k=0}^{t-1} A^k B x_t\right) && \text{by Definition 4.2.1} \\
&\leq L \sum_{t=1}^{T} \left\| \sum_{k=0}^{t-1} A^k B x_{t-k} - \sum_{k=0}^{t-1} A^k B x_t \right\| && \text{by Assumption A4} \\
&\leq \frac{T}{S} L \underbrace{\sum_{t \in T_N} \left\| \sum_{k=0}^{t-1} A^k B x_{t-k} - \sum_{k=0}^{t-1} A^k B x_t \right\|}_{(c)},
\end{aligned}$$

where the last inequality follows because of the following. Consider rounds $t_1 = b_1 S + r$ and $t_2 = b_2 S + r$ for $b_1 < b_2$ and $r \in [S]$. Then, $\|h_{t_1} - \sum_{k=0}^{t_1-1} A^k B x_{t_1}\| \leq \|h_{t_2} - \sum_{k=0}^{t_2-1} A^k B x_{t_2}\|$ because the latter sums over more terms in its history and decisions in consecutive batches have distance bounded above by ${}^{\eta\tilde{L}}/_{\alpha}$ (Theorem 2.1). Therefore, it suffices to show that term (c) is upper bounded by ${}^{\eta\tilde{L}H_1}/_{\alpha}$. We have

$$\begin{aligned}
\sum_{t \in T_N} \left\| \sum_{k=0}^{t-1} A^k B x_{t-k} - \sum_{k=0}^{t-1} A^k B x_t \right\| &\leq \sum_{t \in T_N} \sum_{k=0}^{t-1} \left\| A^k B x_{t-k} - A^k B x_t \right\| \\
&\leq \sum_{t \in T_N} \sum_{k=0}^{t-1} \|A^k\| \|B\| \|x_{t-k} - x_t\| \\
&\leq \sum_{t \in T_N} \sum_{k=0}^{t-1} \|A^k\| \|x_{t-k} - x_t\| && \text{by Assumption A2.}
\end{aligned}$$

Since the same decision x_n is chosen in all rounds of batch n, we can reindex and rewrite

$$\begin{aligned}
\sum_{t\in T_N}\left\|\sum_{k=0}^{t-1} A^k B x_{t-k} - \sum_{k=0}^{t-1} A^k B x_t\right\| &\leq \sum_{t\in T_N}\sum_{k=0}^{t-1} \|A^k\|\|x_{t-k} - x_t\| \\
&\leq \sum_{o=0}^{S-1}\sum_{n=1}^{N-1}\sum_{s=1}^{S} \|A^{(N-n-1)S+s+o}\|\|x_N - x_n\| \\
&\leq \eta\frac{\tilde{L}}{\alpha}\sum_{o=0}^{S-1}\sum_{n=1}^{N-1}\sum_{s=1}^{S}(N-n)\|A^{(N-n-1)S+s+o}\| \\
&= \eta\frac{\tilde{L}}{\alpha}\sum_{o=0}^{S-1}\sum_{n=1}^{N-1}\sum_{s=1}^{S} n\|A^{(n-1)S+s+o}\|,
\end{aligned}$$

where the last inequality follows from bounding the distance between decision in consecutive batches Theorem 2.1 and the triangle inequality. Expanding the triple sum yields

$$\begin{aligned}
&\sum_{o=0}^{S-1}\sum_{n=1}^{N-1}\sum_{s=1}^{S} n\|A^{(n-1)S+s+o}\| \\
&\leq \|A\| + \cdots + \|A^S\| + 2\|A^{S+1}\| + \cdots + 2\|A^{2S}\| + 3\|A^{2S+1}\| + \cdots + 3\|A^{3S}\| + \ldots \\
&\quad + \|A^2\| + \cdots + \|A^{S+1}\| + 2\|A^{S+2}\| + \cdots + 2\|A^{2S+1}\| + 3\|A^{2S+2}\| + \cdots + 3\|A^{3S+1}\| + \ldots \\
&\quad \vdots \\
&\quad + \|A^S\| + \cdots + \|A^{2S-1}\| + 2\|A^{2S}\| + \cdots + 2\|A^{3S-1}\| + 3\|A^{3S}\| + \cdots + 3\|A^{4S-1}\| + \ldots,
\end{aligned}$$

where each line above corresponds to a value of $o \in \{0, \ldots, S-1\}$. Adding up these terms yields H_1. This completes the proof. ■

Note that Theorem 4.10 only provides an upper bound on the policy regret for the general case. Unlike Algorithm 4, it is unclear how to obtain a stronger bound depending on H_p for the case of linear sequence dynamics with the ξ-weighted p-norm for $p > 1$. The above proof can be specialized for this special case, similar to the proofs of Theorem 4.1 and Lemma 4.10.1, to obtain

$$\sum_{t\in T_N}\left\|\sum_{k=0}^{t-1} A^k B x_{t-k} - \sum_{k=0}^{t-1} A^k B x_t\right\| \leq \eta\frac{\tilde{L}}{\alpha}\sum_{o=0}^{S-1}\left(\sum_{n=1}^{N-1}\sum_{s=1}^{S}\left(n\|A^{(n-1)S+s+o}\|\right)^p\right)^{\frac{1}{p}}$$

and

$$\begin{aligned}
&\sum_{o=0}^{S-1}\left(\sum_{n=1}^{N-1}\sum_{s=1}^{S}\left(n\|A^{(n-1)S+s+o}\|\right)^p\right)^{\frac{1}{p}}\\
&\leq \left(\|A\|^p + \cdots + \|A^S\|^p + 2^p\|A^{S+1}\|^p + \ldots 2^p\|A^{2S}\|^p + 3^p\|A^{2S+1}\|^p + \ldots\right)^{\frac{1}{p}}\\
&\quad + \left(\|A^2\|^p + \cdots + \|A^{S+1}\|^p + 2^p\|A^{S+2}\|^p + \ldots 2^p\|A^{2S+1}\|^p + 3^p\|A^{2S+2}\|^p + \ldots\right)^{\frac{1}{p}}\\
&\quad \vdots\\
&\quad \left(\|A^S\|^p + \cdots + \|A^{2S-1}\|^p + 2^p\|A^{2S}\|^p + \ldots 2^p\|A^{3S-1}\|^p + 3^p\|A^{3S}\|^p + \ldots\right)^{\frac{1}{p}}.
\end{aligned}$$

The above expression cannot be easily simplified to $O(H_p)$. However, for the special case of OCO with finite memory, which follows linear sequence dynamics with the 2-norm, we can do so by leveraging the special structure of the linear operator $A_{\text{finite},m}$.

Theorem 4.11. *Consider an online convex optimization with finite memory problem with constant memory length m specified by $(\mathcal{X}, \mathcal{H} = \mathcal{X}^m, A_{finite,m}, B_{finite,m})$. Let the regularizer $R : \mathcal{X} \to \mathbb{R}$ be α-strongly-convex and satisfy $|R(x)-R(\tilde{x})| \leq D$ for all $x, \tilde{x} \in \mathcal{X}$. Algorithm 5 with batch size m and step-size $\eta = \sqrt{\frac{\alpha m D}{T\tilde{L}\left(Lm^{\frac{1}{2}}+\tilde{L}\right)}}$ satisfies*

$$\mathsf{Reg}_T^{\mathsf{OCO-UM}}(\mathsf{Mini-Batch\ FTRL}) \leq O\left(\sqrt{\frac{D}{\alpha}TL\tilde{L}m^{\frac{3}{2}}}\right) \leq O\left(m\sqrt{\frac{D}{\alpha}TL^2}\right).$$

Furthermore, the decisions x_t switch at most $\frac{T}{m}$ times.

Proof. Given the proof of Theorem 4.10 and the above discussion, it suffices to show that

$$\sum_{o=0}^{S-1}\left(\sum_{n=1}^{N-1}\sum_{s=1}^{S}\left(n\|A^{(n-1)S+s+o}\|\right)^2\right)^{\frac{1}{2}} \leq H_2 = m^{\frac{3}{2}}.$$

Recall that $\|A^k_{\text{finite}}\| = 1$ if $k \leq m$ and 0 otherwise. Using this and $S = m$, we have that the above sum is at most $\sqrt{m} + \sqrt{m-1} + \cdots + \sqrt{1} = O\left(m^{\frac{3}{2}}\right)$. This completes the proof. ■

Part II

Analyzing Interactions Offline via Network Science

CHAPTER 5

RETRIEVING TOP WEIGHTED TRIANGLES IN GRAPHS

In the previous part of the dissertation, we studied how a learner should make decisions while interacting with an environment through the perspective of online decision-making. In this part of the dissertation, we change gears and study how a learner can analyze past interactions offline *after* they have occurred through the perspective of network science. In many applications past interactions can be projected onto a weighted graph. For example, consider a user "listening session" on the music streaming platform Spotify [Kumar et al., 2020]. One possible projection of this data is the following: the nodes are songs, and there is a weighted edge between two songs with the weight equal to the number of times the two songs have co-appeared in a listening session. After building such graphs from interaction data, the learner can get insights into their rich structure by using tools from network science.

Pattern counting in graphs is a fundamental primitive for many network analysis tasks and a number of methods have been developed for scaling subgraph counting to large graphs. Many real-world networks, such as the one described above, carry a natural notion of strength of connection between nodes, which are often modeled by a weighted graph. However, existing scalable graph algorithms for pattern mining are designed for unweighted graphs. In this chapter we develop a suite of deterministic and randomized sampling algorithms that enable the fast discovery of the 3-cliques (triangles) with the largest weight in a graph, where weight is measured by a generalized mean of a triangle's edge weights. For example, one of our proposed algorithms can find the top-1000 weighted triangles of a weighted graph with billions of edges (the Spotify graph from above) in thirty seconds on a commodity server. This is orders of magnitude faster than existing "fast" enumeration schemes. Our methods thus open the door towards scalable pattern mining in weighted graphs and provide new tools for analyzing interactions offline. This chapter is based on joint work with Paul Liu, Moses Charikar, and Austin Benson [Kumar et al., 2020].

5.1 Introduction

Small subgraph patterns, also called graphlets or network motifs, have proven fundamental for the understanding of the structure of complex networks [Milo et al., 2002, 2004, Benson et al., 2016]. One of the simplest non-trivial subgraph patterns is the triangle (3-clique), and the basic problem of triangle counting and enumeration has been studied extensively from theoretical and practical perspectives [Avron, 2010, Eden et al., 2017, Kolda et al., 2013, Berry et al., 2015, Stefani et al., 2017a]. These developments are often driven by the desire to scale graph counting to large networks, where performing computations naively is intractable. The focus on triangles is in part spurred by the widespread use of the pattern in graph mining applications, including community detection [Berry et al., 2011, Gleich and Seshadhri, 2012, Rohe and Qin, 2013], network comparison [Contractor et al., 2006, Mahadevan et al., 2007, Pržulj, 2007], representation learning [Henderson et al., 2012, Rossi and Ahmed, 2015], and generative modeling [Robins et al., 2007, Robles-Granda et al., 2016]. Additionally, triangle-based network statistics such as the clustering coefficient are used extensively in the social sciences [Durak et al., 2012, Lawrence, 2006, Burt, 2007, Welles et al., 2010].

Nearly all of the algorithmic literature on scalable counting or enumeration of triangles focuses on *unweighted* graphs. However, many real-world network datasets have a natural notion of *weight* attached to the edges of the graph [Barrat et al., 2004]. For example, edge weights can capture tie strength in social networks [Wasserman and Faust, 1994], traffic flows in transportation networks [Jia et al., 2019], or co-occurrence counts in projections of bipartite networks [Xu et al., 2014]. Such edge weights offer additional insight into the structure of these networks. Moreover, edge weights can enrich the types of small subgraph patterns that are used in analysis. For instance, the network clustering coefficient has been generalized to account for edge weights [Opsahl and Panzarasa, 2009, Onnela et al., 2005b]; in these cases, a triangle is given a weight derived from the weights of its constituent edges. Roughly speaking, the weight of a triangle is typically some combination of the arithmetic mean, geometric mean, minimum and maximum of the edge weights of the triangle. All this being said, we still lack the algorithmic tools for fast analysis of modern large-scale weighted networks, especially in the area of weighted triangle listing and counting.

In applications of weighted triangles in this big data regime, it can often suffice to retrieve only the k triangles of largest weight for some suitable k. For example, in large online social networks, the weight of an edge could reflect how likely it is for users to communicate with each other, and top weighted triangles and cliques in this network could be used for group chat recommendations. In such a scenario, we would typically only be interested in a small number of triangles whose nodes are very likely to communicate with each other as opposed to finding *all* triangles in the graph.

Another application for finding top-weighted triangles appears in prediction tasks involving higher-order network interactions. The goal of the "higher-order link prediction" problem is to predict which new groups of nodes will simultaneously interact (such as which group of authors will co-author a scientific paper in the future, which group of songs will co-appear in a listening session in the future, etc.) [Benson et al., 2018]. In this setting, existing algorithms first create a weighted graph where an edge weight is the number of prior interactions that involve the two end points. Then, they predict that the top-weighted triangles in this weighted graph will appear as higher-order interactions in the future. (Here, weight is measured by a generalized mean of the triangle's edge weights.) Again, it is not necessary to find all triangles since only the top predictions will be acted upon. Existing triangle enumeration algorithms do not scale to massive graphs for these problems. Therefore, we need efficient algorithms for retrieving triangles in large weighted graphs.

In this work we address the problem of enumerating the top-weighted triangles in a weighted graph. Formally, let $G = (V, E, w)$ be a simple, undirected graph with strictly positive edge weights w. Let the weight of a triangle in G be the generalized p-mean of its consitutent edge weights. That is, if a triangle defined by vertices a, b, and c has edge weights w_{ab}, w_{bc}, and w_{ac}, then the weight of the triangle is

$$m_p(a,b,c) = \left(\frac{1}{3}(w_{ab}^p + w_{bc}^p + w_{ac}^p)\right)^{\frac{1}{p}}. \tag{5.1}$$

Given G and an integer parameter k, we develop algorithms to extract the *top-k triangles* in G. We use top-k to refer to the triangles having the k largest weights or, in other words, the k-heaviest triangles. Note that some special cases of the p-mean include arithmetic mean ($p = 1$), geometric mean ($p = 0$), harmonic mean ($p = -1$), minimum ($p = -\infty$) and maximum ($p = \infty$). This family of means is more general and includes those previously examined by Opsahl and Panzarasa [2009] and Benson et al. [2018].

At a high level, we develop two families of algorithms for extracting top-weighted triangles. The first family of algorithms (Section 5.3) is deterministic and optimized for extracting top-k weighted triangles for small k (typically up to a few tens of thousands). These algorithms take advantage of the inherent heavy-tailed edge weight distribution common in real-world networks. In the most general case we show that under a modified configuration model, these algorithms are even "distribution-oblivious". That is, they can automatically compute optimal hyper-parameters for the algorithm for a wide range of input graph distributions. Additionally, the algorithmic analysis is done in a continuous sense (rather than discrete), which may be of independent interest. The second family of algorithms (Section 5.4) is randomized and aims to extract a large number of heavy triangles (not necessarily the top-k). We show that this family of sampling algorithms is closely connected to the prior sampling algorithms for *counting* triangles on *unweighted* graphs [Seshadhri et al., 2014]. Furthermore, we show that these sampling algorithms are easily parallelizable.

We find that a carefully tuned parallel implementation of our deterministic algorithm performs well across a broad range of large weighted graphs, even outperforming the fast randomized sampling algorithms that are not guaranteed to enumerate all of the top-weighted triangles. A parallel implementation of our algorithm running on a commodity server with 64 cores can find the top 1000 weighted triangles in under 10 seconds on several graphs with hundreds of millions of weighted edges and in 30 seconds on a graph with nearly two billion weighted edges. We compare this with the off-the-shelf alternative approach, which would be an intelligent triangle enumeration algorithm that maintains a heap of the top-weighted triangles. Our proposed algorithms are orders of magnitude faster than this standard approach.

5.2 Additional Related Work

Due to its wide applicability, there is a plethora of work on *unweighted* triangle-related algorithms. In the context of enumeration algorithms, recent attention has focused on enumeration in the distributed and parallel setting [Chu and Cheng, 2011, Suri and Vassilvitskii, 2011, Arifuzzaman et al., 2013, Rahman and Hasan, 2013]. These algorithms typically employ an optimized brute force method on each machine [Latapy, 2008, Berry et al., 2010] and the main algorithmic difficulty is in deciding how to partition the data amongst the machines. Although each machine employs a brute force algorithm, early research shows that these algorithms run in time almost linear in the number of edges so long as the *degeneracy* of the graph is small [Chiba and Nishizeki, 1985]. This has led to efficient enumeration algorithms [Berry et al., 2015, Suri and Vassilvitskii, 2011]. For comparison with our methods, we modify such a fast enumeration method (specifically NodeIterator++ [Suri and Vassilvitskii, 2011]) to retain the top-k weighted triangles. Although enumeration algorithms are agnostic to edge weights, we note that the sheer number of triangles in massive graphs renders such an approach prohibitively expensive.

When *enumeration* becomes intractable, triangle-related algorithms focus instead on merely triangle *counts* or graph statistics such as clustering coefficients. Again, these statistics are in the unweighted regime as only the number of triangles is considered. There is a progression of sampling methods depending on what kind of structures one is sampling from the graph. At a basic level, edge-based sampling methods sample an edge and count all incident triangles on that edge. So-called wedge-based methods sample length-2 paths [Kolda et al., 2013] and this concept has been generalized for counting 4-cliques [Jha et al., 2015]. Finally, tiered-sampling combines sampling of arbitrary subgraphs to count the occurrence of larger graphs (with a focus on 4-cliques and 5-cliques) [Stefani et al., 2017b].

Beyond enumeration and sampling, there are numerous other methods for triangle-based algorithms, such as graph sparsification methods [Tsourakakis et al., 2009, Pagh and Tsourakakis, 2012, Etemadi et al., 2016], spectral and matrix based methods [Tsourakakis, 2008, Alon et al., 1997], and a multitude of methods for computing clustering and closure coefficients [Rahman and Hasan, 2014, Seshadhri et al., 2014, Schank and Wagner, 2005, Yin et al., 2019]. For a deeper background on triangle counting, we refer the reader to the overview of Hasan and Dave [2018].

All of the above methods are for triangles. These ideas have been extended in several ways. There are parallel clique enumeration methods [Danisch et al., 2018], and sampling methods for estimating counts of more general motifs [Ahmed et al., 2014, Jain and Seshadhri, 2017, Bressan et al., 2017] and motifs with temporal structure [Liu et al., 2019]. Still, these methods do not work for weighted graphs, where subgraph patterns appear in generalizations of the clustering coefficient [Onnela et al., 2005a] as well as in graph decompositions [Soufiani and Airoldi, 2012].

5.3 Deterministic Algorithms

In this section we develop two types of deterministic algorithms for finding the top-k weighted triangles in a graph. Recall that the weight of a triangle is given by the generalized p-mean of its edge weights (Eq. (5.1)). A simple and robust baseline algorithm is to employ a fast triangle enumeration algorithm for unweighted graphs, compute the weight of each triangle, and pick out the top-k weighted triangles. (To save memory we can also maintain a heap of the top-weighted triangles instead.) In our experiments we use an optimized sequential version of NodeIterator++[Berry et al., 2015, Suri and Vassilvitskii, 2011, Chiba and Nishizeki, 1985], which is the basis for many parallel enumeration algorithms. We call this a "brute-force" approach. We note that there faster, parallel versions of this approach. However, as the results in Section 5.5 show, even a brute force enumerator with perfect parallelism would require over 2000 cores (or machines) to beat our sequential deterministic algorithm in certain cases.

The brute force approach is agnostic to the distribution of edge weights. However, intuitively we expect that triangles of large weight are formed by edges of large weight. We exploit this intuition below to develop faster algorithms. At a high level, our main deterministic algorithm dynamically partitions edges into "heavy" and "light" classes based on edge weight. Following this partition, it finds triangles adjacent to the heavy edges until the top-k heaviest are identified.

5.3.1 A Simple Heavy-light Algorithm

As a precursor to our dynamic algorithm, consider a simple threshold-based algorithm (Algorithm 6) as follows. Given a threshold τ, partition the edges into a "heavy" set $H = \{e \mid w_e > \tau\}$ and a "light" set $L = E \setminus H$. For a large threshold τ, we expect most of the edges to be in the "light" class. Thus, the subgraph $G[H]$ induced by H is small and we may run any enumeration algorithm on it to get a collection of heavy triangles. This is not by itself guaranteed to find the heaviest triangles—edges in H might only appear in triangles with edges in L. However, we find that in practice many of the heaviest triangles have all of their edges in H. We note that with no additional *asymptotic cost*, we can use existing triangle enumeration algorithms to check for triangles with only one or two heavy edges (and the rest light). Unfortunately, the constant factor slowdowns substantially increase the running time on real-world graphs.

Algorithm 6: Static Heavy-light Algorithm.

```
  Input: Weighted graph $G = (V, E, w)$, scaling $p$, number of triangles $k$.
1 Set $H = \{e \in E : w_e > \tau\}$.
2 Set $T = \{$all triangles formed by edges in $H\}$.
3 return k triangles in T with largest p-mean weight.
```

This simple algorithm vastly outperforms the brute force approach in practice and can always find the top-weighted triangles given a proper threshold. Therefore, this method serves as a robust baseline. Nonetheless, there are two issues with the static heavy-light algorithm. First, it relies on a single threshold τ to partition the edges into the light and heavy sets. This leads to enumerating many more triangles than necessary. Second, it is difficult to know what an appropriate value for τ may be given no prior knowledge of the input graph. In the rest of this section we develop a dynamic variant to deal with these issues.

5.3.2 A Dynamic Heavy-light Algorithm

In the rest of Section 5.3 we use w_i to refer to the weight of the i-th heaviest edge. This is in contrast to our usual notation of using w_i to denote the weight of edge i. The meaning should be clear from the context.

To understand the motivation behind our dynamic heavy-light algorithm (Algorithm 7), consider the following. Suppose we preprocess the edges and sort them by *decreasing* weight. That is, $E = \{e_0, e_1, \dots, e_{m-1}\}$, where $w_0 \geq w_1 \geq \dots w_{m-1}$. Consider a partition of the edges into *three* sets based on edge weight:

- $S = \{e_0, \dots, e_h\}$, i.e., "super-heavy" edges that have the $h+1$ largest weights.
- $H = \{e_{h+1}, \dots, e_l\}$, i.e., "heavy" edges that have the next $l-h$ largest weights.
- $L = \{e_{l+1}, \dots, e_{m-1}\}$, i.e., the remaining "light" edges that are neither "heavy" nor "super-heavy".

(As we explain later, Algorithm 7 adjusts this partition dynamically by changing the indices h and l; hence the name "dynamic" heavy-light.) Any triangle can be broken down into a combination of "super-heavy", "heavy", and "light" edges.

As a first order approximation, we intuitively expect the heaviest class of triangles to have three super-heavy edges, the next heaviest to have two super-heavy edges and one heavy edge, and so on down to the case of three light edges. Furthermore, if we consider edges and enumerate triangles in a specific order, then we can obtain an upper bound on the weight of the heaviest triangle we have not yet enumerated. For example, suppose we have enumerated all triangles containing three super-heavy edges. Then the heaviest triangle not yet enumerated must have at least one edge from either H or L. This upper bounds the p-th power of the weight of that triangle to be $\frac{1}{3}\left(2w_0^p + w_h^p\right)$. Algorithm 7 dynamically adjusts the partition of the edges and enumerates triangles in a way so that this bound decreases as quickly as possible.

The while loop in Algorithm 7 runs until the algorithm enumerates k triangles above a dynamically decreasing threshold $\tau = w_h^p + 2w_l^p$. Each iteration of the while loop consists of two steps: (i) updating the partition by moving an edge to a heavier class; and (ii) enumerating triangles whose edges come from certain classes. The constant α_p determines how edges get promoted to heavier classes. We will specify this constant later in our analysis, where we use it to optimize the expected decrease in the threshold τ. At the end of each iteration, the algorithm maintains a loop invariant that it has enumerated all triangles with at least one super-heavy edge or at least two heavy edges. This invariant is what allows us to obtain a bound τ on the heaviest triangle not yet enumerated. We now prove these properties formally and show that Algorithm 7 correctly returns the top-k triangles provided the graph has at least k triangles.

Algorithm 7: Dynamic Heavy-light Algorithm.

```
Input: Weighted graph $G = (V, E, w)$, scaling $p$, number of triangles $k$,
       parameter $\alpha_p$.
1  Sort $E$ in decreasing order of weight.
2  Set threshold $\tau = \infty$.
3  Set triangle set $T = \emptyset$.
4  Set partitions $S = H = \emptyset$ and $L = E$.
5  Set edge pointers $h = l = -1$.
   // We take the convention that $e_{-1} = \infty$.
6  while there are $< k$ triangles above weight $\tau$ in $T$ do
7      if $w_{l+1}^{\alpha_p} > w_{h+1}$ then
8          Move $e_{l+1}$ from $L$ to $H$.
9          $Y$ = triangles formed by $e_{l+1}$ and 2 edges from $S \cup H$.
10         $Z$ = triangles formed by $e_{l+1}$, 1 edge from $L$ and 1 edge from $S \cup H$.
11         $T = T \cup (Y \cup Z)$.
12         $l = l + 1$.
13     else
14         Move $e_{h+1}$ from $H$ to $S$.
15         $Y$ = triangles formed by $e_h$ and 2 edges from $L$.
16         $T = T \cup Y$.
17         $h = h + 1$.
18     end
19     Update threshold $\tau = w_h^p + 2w_l^p$.
20 end
21 return k triangles in T with largest p-mean weight.
```

Lemma 5.3.1. *Algorithm 7 maintains the loop invariant that it has enumerated all triangles with at least one super-heavy edge or at least two heavy edges.*

Proof. The invariant is true before the while loop starts: all edges are light, so there are no triangles with at least one super-heavy edge or at least two heavy edges. Suppose the invariant is true at the start of an iteration of the while loop. We consider two cases.

Suppose $w_{l+1}^{\alpha_p} > w_{h+1}$. Then, edge e_{l+1} moves from L to H (Line 7), and all triangles including e_{l+1} and at least one edge from $S \cup H$ are enumerated. Combining this with the assumption that the invariant was true at the start of the iteration, we have that all triangles with at least one super-heavy edge or at least two heavy edges are enumerated.

Suppose $w_{l+1}^{\alpha_p} \leq w_{h+1}$. Then, edge e_{h+1} moves from H to S (Line 13), and all triangles including e_{h+1} and two edges from L are enumerated. Again, combining this with the assumption that the invariant was true at the start of the iteration, we have that all triangles with at least one super-heavy edge or at least two heavy edges are enumerated. ∎

Lemma 5.3.2. *Algorithm 7 maintains the loop invariant that the threshold $\tau = w_h^p + 2w_l^p$ is an upper bound on the weight of the heaviest triangle not yet enumerated.*

Proof. Consider a triangle Δ with weight at least $w_h^p + 2w_l^p$. By case analysis, there must either exist one edge with weight at least w_h or two edges with weight at least w_l. This means that either one edge is from S or two edges are from H. In either case, Lemma 5.3.1 ensures that Δ must have been enumerated. In fact, a similar reasoning shows that a tight threshold is $w_{h+1}^p + w_{l+1}^p + w_{l+2}^p$ because the subgraph consisting of edges $e_{h+1}, e_{l+1}, e_{l+2}$ is potentially an unenumerated triangle. However, this sum is at most $w_h^p + 2w_l^p$ due to the monotonicity of the edge weights. Therefore, $\tau = w_h^p + 2w_l^p$ is an upper bound on the weight of the heaviest triangle not yet enumerated. ∎

Corollary 5.3.1. *When Algorithm 7 terminates, T contains the top-k weighted triangles in the graph.*[1]

Proof. By Lemma 5.3.2, the heaviest triangle not yet enumerated has weight at most τ. By the algorithm's termination condition, there are at least k triangles in T with weight at least τ. Therefore, the top-k triangles in the graph are enumerated and must be in T. ■

Combining the above three claims, we have that Algorithm 7 correctly returns the top-k triangles provided the graph has at least k triangles. The dynamic heavy-light algorithm solves both issues of the static heavy-light algorithm. Let τ^* denote the weight of the k-th heaviest triangle. As soon as $\tau \leq \tau^*$, the algorithm will have enumerated all the top-k triangles. If the threshold τ hits τ^* exactly, the algorithm only enumerates around k triangles. As τ is computed on the fly, there is no need to choose the threshold at which we partition the edges. In fact, in the next subsection we also provide a parameter-free version of our algorithm where α_p is computed *implicitly*. The algorithmic analysis is done in a continuous sense (rather than discrete), which may be of independent interest.

[1]Note that there may be triangles not enumerated with weight *equal* to one of the triangles in T.

5.3.3 Analysis of the Dynamic Heavy-light Algorithm

We ended the previous subsection with a proof of correctness for the dynamic heavy-light algorithm. We showed that it *always* returns the top-k weighted triangles provided the graph has k triangles. However, there are a few unresolved questions. Where does the if condition on Line 7 of Algorithm 7 come from? How should one set the parameter α_p? Is it possible to remove the need to set this parameter, i.e., have a parameter-free version of the dynamic heavy-light algorithm? In this subsection we provide a simple analysis that sheds light on these questions.

Although Algorithm 7 is a discrete-time algorithm, we analyze it using continuous differentials. Think of time t as a continuous counter for the total elapsed computation time. Let $w_h(t)$ and $w_l(t)$ denote the weight of the edges e_h and e_l at time t respectively. Then, the threshold at time t is $\tau(t) = w_h(t)^p + 2w_l(t)^p$. As a proxy to maximizing the triangles enumeration rate, we would like to maximize the rate at which this threshold decreases. Consequently, we would like to maximize the derivative $\mathrm{d}\tau/\mathrm{d}t$ by adjusting $w_h(t)$ or $w_l(t)$. Note that

$$\mathrm{d}\tau/\mathrm{d}t = pw_h(t)^{p-1}\mathrm{d}w_h/\mathrm{d}t + 2pw_l(t)^{p-1}\mathrm{d}w_l/\mathrm{d}t. \tag{5.2}$$

The derivatives $\mathrm{d}w_h/\mathrm{d}t$ and $\mathrm{d}w_l/\mathrm{d}t$ approximate the change in w_h and w_l per "unit of computation time". In each iteration of the while loop we can choose to spend time decreasing w_h or w_l. Thus, a rough approximation to the derivatives is the ratio of the change in weight (by incrementing either the h or l pointer) to the computational cost of changing the corresponding pointer.

Suppose the edge weights come from a probability distribution. Let $\mathsf{CDF}(w)$ and $\mathsf{PDF}(w)$ denote the cumulative distribution function and probability density function of the edge weight distribution respectively. Let $w_{<h} = \max\{w : w < w_h\}$ denote the largest weight strictly less than w_h. If we move the edge pointer h, then the average change in w_h is the ratio of $(w_h - w_{<h})$ to the number of edges that have weight w_h. The number of edges that have weight w_h is proportional to $\mathsf{CDF}(w_h) - \mathsf{CDF}(w_{<h})$. So, the (average) change in w_h is approximately

$$\frac{w_h - w_{<h}}{CDF(w_h) - CDF(w_{<h})} \approx \frac{1}{PDF(w_h)}. \tag{5.3}$$

Similarly, the (average) change in w_l is approximately $\frac{1}{PDF(w_l)}$.

To recap our analysis so far, we would like to maximize $\mathrm{d}\tau/\mathrm{d}t$, which depends on the derivatives $\mathrm{d}w_h/\mathrm{d}t$ and $\mathrm{d}w_l/\mathrm{d}t$ (Eq. (5.2)). These derivatives are approximately the change in the weight per unit of computation time. In turn, these derivatives are approximately the product of two terms: the inverse of the PDF of the edge weight distribution and the inverse of the computation time required to move the corresponding pointer.

In what follows, we will first analyze the dynamic heavy-light algorithm under some assumptions about the graph generation process and the edge distribution. Later in this subsection, we will use the insights from the analysis to develop a parameter-free and distribution-oblivious version of the algorithm. So, for now, we make the following assumptions:

A1 The edge weights follow a power law distribution. Formally, let W be a random variable. We say that W follows a power law distribution with parameter $\beta >$ and some constant $a > 0$ if $\mathbf{Pr}[W \geq w] \sim aw^{1-\beta}$ for large w. Thus, for large w, $\mathsf{CDF}(w) = O(w^{1-\beta})$ and $\mathsf{PDF}(w) = O(w^{-\beta})$. Not only is this an important case that is easily analyzable, we find that this is a reasonable model for several of our datasets. See Fig. 5.1 for examples.

A2 The graph G is generated using the following simple configuration model [Newman et al., 2001]. Each node v draws its degree d_v from a univariate degree distribution with the sum of degrees being even. The graph is generated from the following random process. Each node v starts out with d_v stubs. While there are stubs available, two random stubs are drawn from the set of all stubs, and the nodes corresponding to those stubs are connected. Furthermore, upon connection a random edge weight drawn from the edge weight distribution is assigned to the edge. In the end, all self-loops in the graph are discarded. While this assumption is quite strong, we find that this simple configuration model yields good estimates for optimal values of α_p in practice. See the experimental results in Section 5.5.

Recall from above that the (average) change in w_h and w_l can be written as $\frac{1}{\mathsf{PDF}(w_h)}$ and $\frac{1}{\mathsf{PDF}(w_l)}$ respectively. Assumption **A1** says that $\mathsf{PDF}(w) = O(w^{-\beta})$. Using this we can write the (average) change in w_h and w_l as $O(w_h^{\beta})$ and $O(w_l^{\beta})$ respectively.

Now we analyze the computational cost of moving the h and l pointers. Let $G_H = G[S \cup H]$ and $G_L = G[L]$ denote the subgraphs induced by the edge sets $S \cup H$ and L respectively. Let $\bar{d}_{G_H}$ and $\bar{d}_{G_L}$ denote their average degree. With appropriate

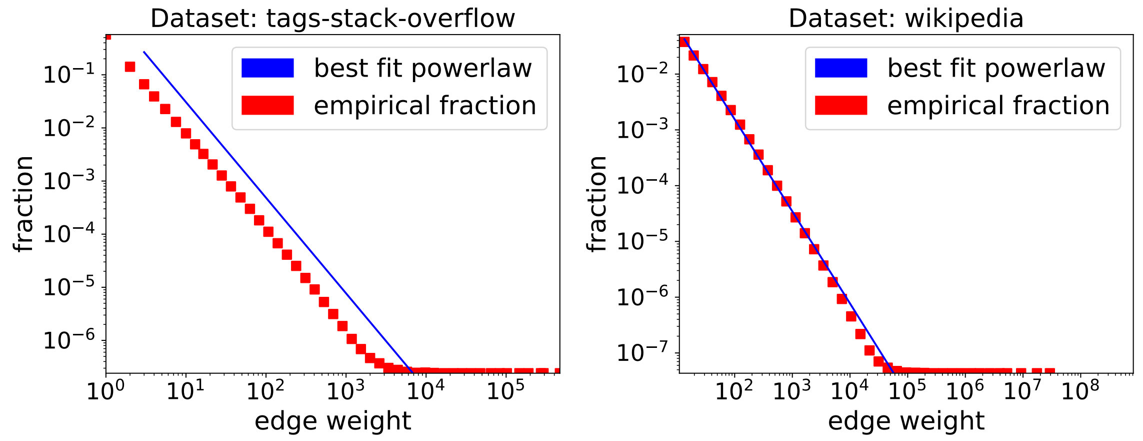


Figure 5.1: Edge weight distribution in two datasets. (See Section 5.5.1 for a description of the datasets). These plots suggest that a power law distribution on edge weights is a reasonable assumption. With this, we have a simple condition (Line 7) to choose which pointer to move in the dynamic heavy-light algorithm (Algorithm 7).

data structures for checking the existence of edges in G_H, the cost of moving the h pointer is bounded by the degree sum of the endpoints of the edge e_h in G_L. On average, this is $O\left(\bar{d}_{G_L}\right)$. If we assume that G_L has approximately as many edges as G, which is valid when k is small, then $\bar{d}_{G_L} \approx \bar{d}_G$. Thus, the computational cost of moving the h pointer is approximately $O\left(\bar{d}_G\right)$. Similarly, the computational cost of moving the l pointer is bounded by the degree sum of the endpoints of the edge e_l in G_H. On average, this is $O\left(\bar{d}_{G_H}\right)$. Since the number of edges in G_H is exactly $|S \cup H|$, assumption **A1** on the edge weight distribution gives us that $\bar{d}_{G_H} = O\left(CDF(w_l)\bar{d}_G\right)$. Thus, the computational cost of moving e_l is approximately $O\left(w_l^{1-\beta}\bar{d}_G\right)$.

Recall that a rough approximation to the derivatives $\mathrm{d}w_h/\mathrm{d}t$ and $\mathrm{d}w_l/\mathrm{d}t$ is the ratio of the change in weight (by incrementing either the h or l pointer) to the computational cost of changing the corresponding pointer. Combining the above analyses for the (average) change in weight and the computational cost of changing a pointer, we have that

$$\mathrm{d}w_h/\mathrm{d}t = O\left(\frac{w_h^{\beta}}{\bar{d}_G}\right) \quad \text{and} \quad \mathrm{d}w_l/\mathrm{d}t = O\left(\frac{w_l^{2\beta-1}}{\bar{d}_G}\right). \tag{5.4}$$

Using this we can rewrite Eq. (5.2) as

$$\begin{aligned}\mathrm{d}\tau/\mathrm{d}t &= pw_h^{p-1}\mathrm{d}w_h/\mathrm{d}t + 2pw_l^{p-1}\mathrm{d}w_l/\mathrm{d}t \\ &= pw_h^{p-1}O\left(\frac{w_h^{\beta}}{\bar{d}_G}\right) + 2pw_l^{p-1}O\left(\frac{w_l^{2\beta-1}}{\bar{d}_G}\right).\end{aligned} \tag{5.5}$$

Now, in each iteration of the dynamic heavy-light algorithm, we can only change one of the h and l pointers. Since w_h and w_l are decreasing as the algorithm progresses, both derivatives are monotonically decreasing. This monotonicity property means that greedily choosing the pointer to increment is optimal. Therefore, we should greedily change the pointer that gives the most "bang per buck", i.e., choose h and l such that

$$\begin{aligned} w_h^{p-1}\mathrm{d}w_h/\mathrm{d}t &= 2w_l^{p-1}\mathrm{d}w_l/\mathrm{d}t \\ \Leftrightarrow w_h^{p-1}O\left(\frac{w_h^{\beta}}{\bar{d}_G}\right) &= 2w_l^{p-1}O\left(\frac{w_l^{2\beta-1}}{\bar{d}_G}\right) \\ \Rightarrow w_h &= O\left(w_l^{2-\frac{p}{p-1+\beta}}\right). \end{aligned} \tag{5.6}$$

In other words, we should maintain the edge pointers h and l such that the weights are separated geometrically by $\alpha_p = 2-\frac{p}{p-1+\beta}$. This answers two of the questions we mentioned at the start of this subsection. The above analysis motivates the choice of the if condition in Line 7 of Algorithm 7 and shows how to set the parameter α_p. This holds under assumptions **A1** and **A2**. Next, we use the above analysis to design a parameter-free and distribution-oblivious version of the dynamic heavy-light algorithm.

Distribution-oblivious dynamic heavy-light algorithm. Now we describe how to modify the dynamic heavy-light algorithm to obtain a "distribution-oblivious" variant in which α_p can be estimated implicitly on the fly, albeit with less robustness in practice than simply setting it. Previously, we assumed that the edge weights follow a power law distribution (Assumption **A1**). However, our analysis holds more generally. In particular, as long as the derivatives $\mathrm{d}w_h/\mathrm{d}t$ and $\mathrm{d}w_l/\mathrm{d}t$ decrease monotonically with time, our greedy strategy of moving the pointer that gives the most "bang per buck" will be optimal. Therefore, we only need to assume that the PDF of the edge weight distribution monotonically increases as the weight decreases. This includes a wide family of distributions, including power law distributions, uniform distributions, etc.

The previous analysis shows that we would like to maximize $\mathrm{d}\tau/\mathrm{d}t$ (Eq. (5.2)). We used Assumptions **A1** and **A2** to derive a closed-form expression for this derivative Eq. (5.5). In the distribution-oblivious setting, we can maintain an estimate of the derivatives $\mathrm{d}w_h/\mathrm{d}t$ and $\mathrm{d}w_l/\mathrm{d}t$ as the algorithm runs and greedily change the pointer with the higher value of $w^{p-1}\mathrm{d}w/\mathrm{d}t$. The derivatives $\mathrm{d}w_h/\mathrm{d}t$ and $\mathrm{d}w_l/\mathrm{d}t$ are approximately the change in weight per unit of computation time.

The change in weight for w_h is estimated by the ratio of $w_h - w_{<h}$ and the number of edges that have weight w_h, and similarly for w_l. The computational cost of changing the h pointer can be estimated by the sum of the degrees of the endpoints of the edge e_h in G_L. Similarly, the computational cost of changing the l pointer can be estimated by the sum of the degrees of the endpoints of the edge e_l with G_H. Consequently, we obtain a "distribution-oblivious" algorithm that works on a family of edge weight distributions whose PDF monotonically increases as weight decreases.

In our experiments Section 5.5 we find that this automatic way of implicitly computing α_p is quite successful. However, sometimes, noise in the derivative estimates cause the algorithm to be slower than using a set value of α_p.

5.4 Randomized Sampling Algorithms

In this section we develop randomized sampling algorithms designed to sample a large collection of triangles with large weight. Formally, given a generalized p-mean as a weight function, these algorithms sample triangles exactly proportional to their weight. The main difference between the algorithms is how efficiently they can generate samples.

In particular, we generalize three types of sampling schemes that have been used to estimate triangle counts in unweighted graphs. The first scheme is based on *edge sampling* [Kolountzakis et al., 2012, Tsourakakis et al., 2011, Pagh and Tsourakakis, 2012], which first samples one edge and then enumerates triangles adjacent to the sampled edge. The second uses ideas from *wedge sampling* [Seshadhri et al., 2014], which samples two adjacent edges and checks whether these two edges induce a triangle. The final approach generalizes the idea of path sampling [Jha et al., 2015], which samples a three-edge path and checks if it induces a triangle. Although these approaches were all designed for triangle counting in unweighted graphs, they generalize quite seamlessly to simple schemes for sampling highly weighted triangles. The main benefits of these algorithms are that they are simple to implement and also easy to parallelize, since samples can be trivially generated in parallel.

Recall that the weight of a triangle is given by a generalized p-mean of its edge weights (Eq. (5.1)). Note that the weighted ordering of triangles is independent of the scaling by $\frac{1}{3}$ and the exponent $\frac{1}{p}$. So, in the rest of this section we consider the simpler weighting function

$$w_p(a, b, c) = w_{ab}^p + w_{bc}^p + w_{ac}^p. \tag{5.7}$$

We will also use the notation $N(a)$ to denote the set of neighbors of a vertex $a \in V$, i.e., $N(a) = \{b \in V \mid (a, b) \in E\}$, and d_a to denote the degree of vertex a, i.e., $d_a = |N(a)|$.

Algorithm 8: Weighted Edge Sampling Algorithm.

```
Input: Weighted graph $G = (V, E, w)$, scaling $p$, number of triangles $k$,
       number of iterations $t$.
1 Set $T = \emptyset$.
2 for iteration $1, \dots, t$ do
3     Sample edge $(a, b) \propto w^p_{ab}$.
4     for each neighbor $c \in N(a) \cap N(b)$ do
5         $T = T \cup \{(a, b, c)\}$.
6     end
7 end
8 return k triangles in T with largest p-mean weight.
```

5.4.1 Weighted Edge Sampling

We start with the first of our randomized sampling algorithms, namely, the weighted edge sampling algorithm (ES) (Algorithm 8). It is based on repeating the following simple two-step procedure over multiple iterations. It first samples a single edge according to the following distribution:

$$\mathbf{Pr}\Big[\text{ sampling edge } (a, b)\Big] = \frac{w^p_{ab}}{Z_{\text{edge}}} \quad \text{with} \quad Z_{\text{edge}} = \sum_{(u,v)\in E} w^p_{uv}.$$

After sampling an edge (a, b), it enumerates all triangles (a, b, c) incident to (a, b). This simple algorithm has the nice property that it samples triangles exactly proportional to their weight.

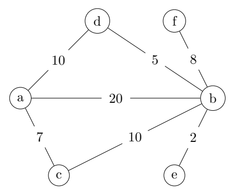
Lemma 5.4.1. *Consider a triangle (a, b, c). In each iteration of Algorithm 8, this triangle is enumerated with probability proportional to its weight, $w_p(a, b, c)$.*

Proof. In each iteration, the probability of sampling edge (a, b) is $\frac{w^p_{ab}}{Z_{\text{edge}}}$, where $Z_{\text{edge}} = \sum_{(u,v)\in E} w^p_{uv}$. Triangle (a, b, c) is enumerated if and only if one of the edges

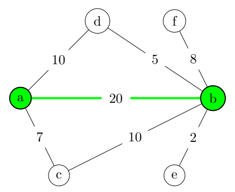
(a, b), (b, c) or (a, c) is sampled. Therefore, the probability of enumerating triangle (a, b, c) is equal to $\frac{w^p_{ab}}{Z_{\text{edge}}} + \frac{w^p_{bc}}{Z_{\text{edge}}} + \frac{w^p_{ac}}{Z_{\text{edge}}} \propto w_p(a, b, c)$. ■

While the weighed edge sampling algorithm is simple to describe, making it fast in practice requires careful implementation. First, a natural way of sampling an edge is to pick one at random with probability proportional to its weight. But this is slow because there are a large number of edges. However, typically there are a much smaller number of *unique* edge weights. So, we first sample an edge weight and then sample an edge with this weight. Second, pre-processing calculations of sampling probabilities for this approach involves iterating over the edges and computing two quantities—a cumulative edge weight (in order to sample an edge weight) and a map of edge weight to edges (in order to sample an edge given an edge weight). This pre-processing step can take much longer than the sampling loop if implemented naively. However, in a sorted list of edges, all edges that share the same edge weight lie in a contiguous chunk. This significantly speeds up the process of computing the above quantities.

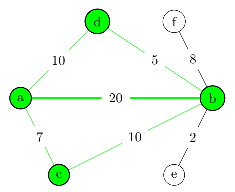
The weighted edge sampling algorithm has a few issues. First, there is no guarantee that it will generate at least k unique triangles. Furthermore, even if it samples a sufficient number of triangles, there is no guarantee that these are the *top-weighted* triangles. This issue is an inherent limitation of randomized sampling algorithms in general. The second issue is that the algorithm takes $O(d_a + d_b)$ time to find triangles adjacent to an edge (a, b). This can be expensive in graphs where high-degree nodes are connected. Our next sampling algorithm mitigates this issue. However, as the experimental results in Section 5.5 show, in practice, the weighted edge sampling algorithm is competitive with the deterministic dynamic heavy-light algorithm (Algorithm 7) and is the most efficient out of our sampling algorithms.



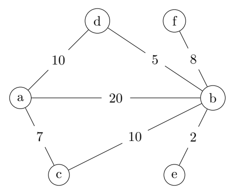
(a) Original graph.



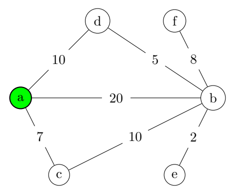
(b) Sample an edge.



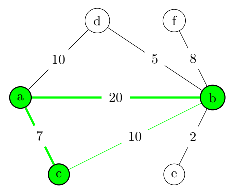
(c) Iterate through neighbors and add triangles to T.



(d) Original graph.



(e) Sample a vertex.



(f) Sample two neighbors and add triangle to T.

Figure 5.2: (top row) Edge sampling: sample an edge, iterate through its neighbors, add triangles. (bottom row): Wedge sampling: sample a vertex, sample two neighbors, add triangle.

5.4.2 Weighted Wedge Sampling

As mentioned above, one of the issues with weighted edge sampling is that it has to iterate over neighbors of the endpoints of the sampled edge to find triangles. This can be expensive if the degrees of the endpoints are large. An alternative approach is to sample adjacent edges—also called *wedges*—with large weight and then check if each wedge induces a triangle. This sampling scheme is called *wedge sampling* and this has been used for estimating the total number of triangles in an unweighted graph [Seshadhri et al., 2014, Türkoglu and Turk, 2017].

The weighted wedge sampling algorithm (Algorithm 9) repeats the following three-step procedure over multiple iterations. It first samples a single node with a bias towards nodes that participate in heavily weighted edges. Specifically, let $D(a) = \sum_{b \in N(a)} w_{ab}^{p}$ denote the sum of the edge weights incident to a. Algorithm 9 samples a node a according to the following distribution:

$$\mathbf{Pr}\Big[\text{ sampling node } a\Big] = \frac{\tilde{W}_1(a)}{Z_{w,1}} = \frac{2 \cdot d_a \cdot D(a)}{Z_{w,1}}, \tag{5.8}$$

where $Z_{w,1}$ is a normalizing constant. Then, it samples a neighbor of node a, again with a bias towards nodes that participate in heavily weighted edges. Specifically, it samples a neighbor of a, say b, according to the following distribution:

$$\mathbf{Pr}\Big[\text{ sampling node } b \in N(a) \mid a\Big] = \frac{\tilde{W}_2(b \mid a)}{Z_{w,2}} = \frac{d_a \cdot w_{ab}^{p} + D(a)}{Z_{w,2}}, \tag{5.9}$$

where $Z_{w,2}$ is a normalizing constant. Finally, it samples a second neighbor of node a, say c, according to the following distribution:

$$\mathbf{Pr}\Big[\text{ sampling node } c \in N(a) \mid a, b\Big] = \frac{\tilde{W}_3(c \mid a, b)}{Z_{w,3}} = \frac{w_{ab}^{p} + w_{ac}^{p}}{Z_{w,3}}, \tag{5.10}$$

where $Z_{w,3}$ is again a normalizing constant. If the sampled wedge $\{(a,b),(a,c)\}$ induces a triangle, then Algorithm 9 adds it to its collection of triangles.

While the exact sampling distributions above may seem arbitrary, they bias the sampling of triangles towards those that are heavily weighted. However, as in the

Algorithm 9: Weighted Wedge Sampling Algorithm

```
Input: Weighted graph $G = (V, E, w)$, scaling $p$, number of triangles $k$,
       number of iterations $t$.
1 Set $T = \emptyset$.
2 for iteration $1, \dots, t$ do
3     Sample node $a$ with probability as in Eq. (5.8).
4     Sample $b \in N(a)$ with probability as in Eq. (5.9).
5     Sample $c \in N(a)$ with probability as in Eq. (5.10).
6     if nodes $a, b, c$ form a triangle then
7         $T = T \cup \{(a, b, c)\}$.
8 end
9 return $k$ triangles in $T$ with largest $p$-mean weight.
```

case of Algorithm 8, they are designed so that each iteration of Algorithm 9 samples a triangle with probability proportional to its weight. In particular, similar to the unweighted wedge sampling scheme of Seshadhri et al. [2014], we show the following.

Lemma 5.4.2. *Consider a triangle (a, b, c). In each iteration of Algorithm 9, this triangle is enumerated with probability $\frac{w_p(a,b,c)}{Z_{wedge}}$, where $Z_{wedge} = \sum_{v \in V} d_v \cdot D(v) = \sum_{v \in V} d_v \sum_{u \in N(v)} w_{uv}^p$.*

Proof. The normalizing constants in Eqs. (5.8) to (5.10) are

$$Z_{w,1} = 2 \sum_{v \in V} d_v \cdot D(v),$$

$$Z_{w,2} = \tilde{W}_1(a),$$

$$Z_{w,3} = \tilde{W}_2(b \mid a).$$

Therefore, the probability of sampling a wedge (a, b, c) centered on node a is equal to

$$\frac{\tilde{W}_1(a)}{Z_{w,1}} \cdot \frac{\tilde{W}_2(b \mid a)}{Z_{w,2}} \cdot \frac{\tilde{W}_3(c \mid a, b)}{Z_{w,3}} = \frac{\tilde{W}_3(c \mid a, b)}{Z_{w,1}} = \frac{w^p_{ab} + w^p_{ac}}{Z_{w,1}}.$$

This implies that the probability of sampling any of the three wedges comprising nodes a, b and c is equal to

$$\frac{w^p_{ab} + w^p_{ac}}{Z_{w,1}} + \frac{w^p_{ac} + w^p_{bc}}{Z_{w,1}} + \frac{w^p_{bc} + w^p_{ab}}{Z_{w,1}} = \frac{w_p(a, b, c)}{Z_{\text{wedge}}},$$

where the last equality follows from the definitions of $Z_{w,1}$ and Z_{wedge}. Triangle (a, b, c) is enumerated if and only if one of its three wedges is sampled. This completes the proof. ■

5.4.3 Weighted Path Sampling

In addition to the above, we can also design a sampling scheme based on path sampling [Jha et al., 2015]. Edge sampling finds triangles after sampling a single edge and wedge sampling finds triangles after sampling two edges; path sampling finds triangles by sampling three edges that are biased towards finding a top-weighted triangle. This sampling scheme performs poorly in practice, but we include it here for the sake of theoretical interest.

The weighted path sampling algorithm (Algorithm 10) repeats the following three-step procedure over multiple iterations. It first samples an edge (a, b). Then it samples two more edges, (a, c) and (b, c'), connected to (a, b) to form a length-three path. If $c = c'$, then this path forms a triangle. Let $\tilde{d}_v := d_v - 1$ and $\tilde{D}_u(v) :=$

Algorithm 10: Weighted Path Sampling Algorithm.

```
Input: Weighted graph $G = (V, E, w)$, scaling $p$, number of triangles $k$,
       number of iterations $t$.
1 Set $T = \emptyset$.
2 for iteration $1, \ldots, t$ do
3     Sample edge $(a, b)$ with probability as in Eq. (5.11).
4     Sample $c \in N(a) \setminus b$ with probability as in Eq. (5.12).
5     Sample $c' \in N(b) \setminus a$ with probability as in Eq. (5.13).
6     if $c = c'$ then
7         $T = T \cup \{(a, b, c)\}$.
8 end
9 return $k$ triangles in $T$ with largest $p$-mean weight.
```

$D(v) - w_{uv}^p$. (Recall that $D(v) = \sum_{u \in N(v)} w_{uv}^p$.) The sampling distributions for these three samples are as follows:

$$\mathbf{Pr}\Big[\text{ sampling edge } (a, b)\Big] = \frac{\tilde{W}_1(a, b)}{Z_{p,1}} = \frac{\tilde{d}_a \tilde{d}_b w_{ab}^p + \tilde{d}_a \tilde{D}_a(b) + \tilde{d}_b \tilde{D}_b(a)}{Z_{p,1}}, \tag{5.11}$$

$$\mathbf{Pr}\Big[\text{ sampling node } c \in N(a) \setminus \{b\} \mid a, b\Big] = \frac{\tilde{W}_2(c \mid a, b)}{Z_{p,2}} = \frac{\tilde{d}_b(w_{ac}^p + w_{ab}^p) + \tilde{D}_a(b)}{Z_{p,2}}, \tag{5.12}$$

$$\mathbf{Pr}\Big[\text{ sampling node } c' \in N(b) \setminus \{a\} \mid a, b\Big] == \frac{\tilde{W}_3(c' \mid a, b)}{Z_{p,3}} = \frac{w_{ab}^p + w_{ac}^p + w_{bc'}^p}{Z_{p,3}}, \tag{5.13}$$

where $Z_{p,1}, Z_{p,2}$ and $Z_{p,3}$ are normalizing constants.

As with weighted wedge sampling, the above sampling distributions may seem arbitrary, but they are designed to sample triangles proportional to their weight.

Lemma 5.4.3. *Consider a triangle (a, b, c). In each iteration of Algorithm 10, this triangle* *is enumerated with probability* $\frac{w_p(a,b,c)}{Z_{path}}$*, where* $Z_{path} = \sum_{u,v \in V} \left(\tilde{d}_u \tilde{d}_v w_{uv}^p + \tilde{d}_u \tilde{D}_u(v) + \tilde{d}_v \tilde{D}_v(u) \right)$.

Proof. The normalizing constants in Eqs. (5.11) to (5.13) are

$$\begin{aligned}
Z_{p,1} &= \sum_{a,b \in V} \left(\tilde{d}_a \tilde{d}_b w_{ab}^p + \tilde{d}_a \tilde{D}_a(b) + \tilde{d}_b \tilde{D}_b(a) \right) \\
&= Z_{\text{path}}, \\
Z_{p,2} &= \sum_{c \in N(a) \backslash b} \tilde{d}_b (w_{ac}^p + w_{ab}^p) + \tilde{D}_a(b) \\
&= \tilde{d}_b \tilde{D}_b(a) + \tilde{d}_a \tilde{d}_b w_a b + \tilde{d}_a \tilde{D}_a(b) \\
&= \tilde{W}_1(a,b), \\
Z_{p,3} &= \sum_{c' \in N(b) \backslash a} w_{ac}^p + w_{ab}^p + w_{bc'}^p, \\
&= \tilde{d}_b (w_{ac} + w_{ab}) + \tilde{D}_a(b) \\
&= \tilde{W}_2(c \mid a,b).
\end{aligned}$$

A triangle (a,b,c) is enumerated if and only if $c = c'$. Therefore, the probability of sampling triangle (a,b,c) is equal to

$$\frac{\tilde{W}_1(a,b)}{Z_{p,1}} \cdot \frac{\tilde{W}_2(c \mid a,b)}{Z_{p,2}} \cdot \frac{\tilde{W}_3(c' \mid a,b)}{Z_{p,3}} = \frac{\tilde{W}_3(c' \mid a,b)}{Z_{p,1}} = \frac{w_p(a,b,c)}{Z_{p,1}} = \frac{w_p(a,b,c)}{Z_{\text{path}}}. \qquad \blacksquare$$

5.4.4 Sample Efficiency

Lemmas 5.4.1 to 5.4.3 say that Algorithms 8 to 10 tend to sample triangles with large weight. However, there is no guarantee that the top-weighted triangles are actually enumerated. The following simple probabilistic analysis gives some insight into the sample efficiency of these algorithms.

Let $q = \frac{w_p(a,b,c)}{Z}$ denote the probability of sampling triangle (a, b, c). (Here, Z depends on the sampling algorithm, and is one of $Z_{\text{edge}}, Z_{\text{wedge}}$ and Z_{path}.) Let s denote the number of samples. Then, $r = 1 - (1 - q)^s$ is the probability that triangle (a, b, c) is sampled at least once in s samples. Using the fact that $1 - x \leq \exp(-x)$, we have that $r \geq 1 - \exp(-sq)$. In other words, let $\delta \in (0, 1)$. Then, if $s \geq \frac{\log(\delta^{-1})}{q} = \frac{Z \log(\delta^{-1})}{w_p(a,b,c)}$, then the probability that triangle (a, b, c) is enumerated is at least $1 - \delta$.

Comparing the expressions for the normalizing constants for the three sampling algorithms, we see that the normalizing constants for weighted wedge and weighted path sampling are larger than for weighted edge sampling. This combined with the above analysis implies that the normalizing constants for weighted wedge and weighted path sampling increase the number of samples required for finding the top-weighted triangles. However, obtaining samples with weighted edge sampling can be more expensive if it has to find common neighbors of nodes with large degree. It is not immediately clear which algorithm is better, but our experimental results in Section 5.5 show that edge sampling is superior in practice.

5.4.5 Extensions to Cliques

All of our sampling algorithms can also be used to sample k-cliques for an arbitrary k. The high level objective of each sampling algorithm is to sample some sort of subgraph with probability proportional to its weight. The three sampling algorithms we presented are ways to sample edges, wedges, and length-3 paths. To convert one of our triangle samplers to a clique sampler, we can sample a subgraph with one of our three algorithms and then enumerate all cliques incident on that subgraph. For edge sampling we can sample an edge and enumerate all $(k-2)$-cliques incident on that edge. For wedge and path sampling we can sample wedges and paths until we sample a triangle; once a triangle is sampled, we can then enumerate all $(k-3)$-cliques incident on that triangle. This natural extension to cliques is an advantage of the sampling algorithms over the dynamic heavy-light algorithm (Algorithm 7), which does not easily extend to larger cliques. Even though our main focus is on triangles and the bulk of our experiments focuses on this setting (Section 5.5), we provide some experiments for cliques in Section 5.6.

5.5 Numerical Experiments

We now report the results of several experiments comparing the performance of our dynamic and randomized sampling algorithms with some competitive baselines, including the static heavy-light algorithm (Algorithm 6). The implementations of our algorithms and scripts to reproduce the results are available at https://github.com/raunakkmr/Retrieving-top-weighted-triangles-in-graphs.

There are two main takeaways. First, our dynamic heavy-light algorithm (Algorithm 7) performs the best across a wide range of datasets, outperforming the baselines by several orders of magnitude in terms of running time. Second, edge sampling (Algorithm 8) performs the best out of our randomized sampling algorithms. While it is competitive with the dynamic heavy-light algorithm, it is not always guaranteed to achieve 100% accuracy. In contrast, the dynamic heavy-light algorithm is usually faster and is always guaranteed to achieve 100% accuracy.

5.5.1 Datasets

We use a number of datasets in order to test the performance our algorithms. The datasets can be found at `http://www.cs.cornell.edu/~arb/data/index.html`. Table 5.1 lists summary statistics of the datasets and we describe them briefly below.

tags-stack-overflow [Benson et al., 2018]. Stack Overflow is a platform where users ask, answer, and discuss computer programming questions. They also annotate questions with 1–5 tags. We construct a graph where the nodes are tags and the weight of an edge is the number of questions jointly annotated by the two tags.

threads-stack-overflow [Benson et al., 2018]. We construct a graph from a dataset of user co-participation on Stack Overflow question threads that last at most 24 hours. Nodes are users, and the weight of an edge is the number of times the two users appeared in the same short question thread.

Wikipedia-clickstream [Wulczyn and Taraborelli, 2017]. We construct a graph from Wikipedia clickstream data (request logs) from January 2017. The logs capture how users transition between articles. (Only transitions appearing at least 11 times were recorded.) The nodes in this graph correspond to Wikipedia articles, and the weight of an edge is the number of times users transitioned between the two pages.

Ethereum Ethereum is a blockchain-based computing platform for decentralized applications. Transactions are used to update state in the Ethereum network, and each transaction has a sender and a receiver address. We construct a graph using data for all transactions on the platform up to August 17, 2018 as provided by `blockchair.com`. The nodes in this graph correspond to addresses, and the weight of an edge is the number of transactions between the two addresses.

AMiner and MAG [Tang et al., 2008, Sinha et al., 2015]. We construct weighted co-authorship graphs from two large bibliographic databases—AMiner and the Microsoft Academic Graph. The nodes in these graphs correspond to authors, and the weight of an edge is the number of papers they have co-authored. Papers with more than 25 authors were omitted from the graph construction.

Table 5.1: Summary statistics of datasets.

dataset	# nodes	# edges	edge weight	
			mean	max
tags-stack-overflow	50K	4.2M	13	469
threads-stack-overflow	2.3M	21M	1.1	546
Wikipedia-clickstream	4.4M	23M	347	817M
Ethereum	38M	103M	2.8	1.9M
AMiner	93M	324M	1.3	13K
reddit-reply	8.4M	435M	1.5	165K
MAG	173M	545M	1.7	38K
Spotify	3.6M	1.9B	8.6	2.8M

reddit-reply [Hessel et al., 2016, Liu et al., 2019]. Users on the social media web site `reddit.com` interact by commenting on each other's posts. We construct a graph from a collection of user comments. Nodes are users, and the weight of an edge is the number of interactions between the two users.

Spotify As part of a machine learning challenge, the music streaming platform Spotify released a large number of user "listening sessions," each consisting of a set of songs. We construct a weighted graph where the nodes represent songs and the weight of an edge is the number of times the songs co-appeared in a session.

5.5.2 Experimental Setup

We evaluate the performance (i.e., the running time) of the following algorithms:

1. Randomized edge sampling (ES) (Algorithm 8).
2. Randomized wedge sampling (WS) (Algorithm 9).
3. Randomized path sampling (PS) (Algorithm 10).
4. Dynamic heavy-light (DHL) (Algorithm 7).
5. Auto heavy-light (AHL), which is the oblivious version of DHL that automatically adjusts edge promotion to optimize the decrease in the threshold. This is described at the end of Section 5.3.3.
6. Static heavy-light (SHL) (Algorithm 6).
7. As a baseline, we use an optimized sequential version of NodeIterator++ [Berry et al., 2015, Suri and Vassilvitskii, 2011, Chiba and Nishizeki, 1985] and refer to this as the "brute force approach" (BF). Essentially, this algorithm iterates over vertices with decreasing degree, and for each vertex it only enumerates triangles that are formed by neighboring vertices with lower degree than itself.

We implement all the algorithms in C++ and run all the experiments on a 64 core 2.20 GHz Intel Xeon CPU with 200 GB of RAM. We use parallel sorting for all algorithms and use parallel sampling for the randomized sampling algorithms. We execute all other parts of the algorithms sequentially. We evaluate the algorithms for two values of k: 1,000 and 100,000. We use the arithmetic mean as the weight of a triangle, i.e., $p = 1$ in Eq. (5.1).

Recall that DHL (Algorithm 7) uses a power law distribution model of the edge weights and sets a parameter α_p based on the power law exponent. We fix $\alpha_p = 1.25$ for our experiments and this works well across a range of datasets.

The randomized sampling algorithms are not guaranteed to enumerate all of the top-k weighted triangles. Instead, we measure the performance of these algorithms in terms of running time and accuracy (the fraction of top-k triangles actually enumerated). We run ES long enough for it to achieve at least 94% ($k = 1,000$) or 50% ($k = 100,000$) accuracy on all datasets. We run WS long enough to achieve 50% accuracy (for both values of k). However, in practice, its performance is poor, and we terminate the algorithm if it takes longer than BF to achieve this accuracy level. PS takes longer than BF to achieve this accuracy level on all datasets, so we do not include it in Table 5.2.

Similarly, SHL is not guaranteed to achieve 100% accuracy because it relies on a fixed threshold to partition the edges as heavy and light, and only enumerates triangles formed by heavy edges. As the threshold decreases a larger number of edges are labelled heavy. This increases the accuracy but also slows down the algorithm. Fig. 5.3 illustrates this trade-off on the Ethereum dataset; a similar trend is observed on the other datasets. In our experiments we label the top 10% of edges as heavy and report the achieved accuracy. As we discuss below, SHL attains sub-100% accuracy in practice and is slower than the other proposed algorithms (DHL and AHL); improving the accuracy would only make this algorithm slower.

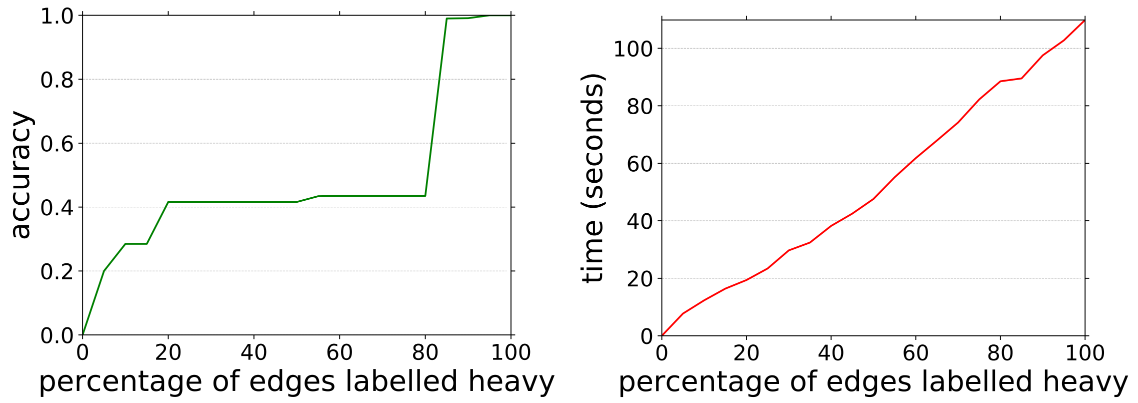


Figure 5.3: Accuracy and running time as a function of edges labeled "heavy" by the thresholding for the static heavy-light algorithm (Algorithm 6) on the Ethereum dataset for $k = 1000$. As the threshold decreases, a larger percentage of edges are labelled heavy. This increases the accuracy but also increases the running time. For reasonable accuracy levels, we find that the running time is slower than our optimized dynamic heavy-light algorithm (Algorithm 7), which achieves 100% accuracy. See Table 5.2.

5.5.3 Results

Table 5.2 shows the running times of all of our algorithms. BF did not terminate on Spotify even after 24 hours, so we ended the execution and indicate the running time as > 86400 seconds. We highlight a few important findings.

1. First, our deterministic algorithms DHL and AHL excel at retrieving the top k triangles. They achieve perfect accuracy while being several orders of magnitude faster than BF. For instance, these algorithms get a 1000x speedup on the reddit-reply dataset ($k = 1,000$) and more than a 2000x speedup on the Spotify dataset ($k = 100,000$). These algorithms also outperform SHL by a significant margin in terms of time and accuracy. For example, despite be-

ing 30x slower on reddit-reply, SHL only achieves 50% accuracy ($k = 1,000$). Again, DHL and AHL *always achieve 100% accuracy*, and do so in a fraction of the time taken by the baseline algorithms BF and SHL.

2. Among the randomized sampling algorithms, ES performs much better than WS. WS struggles to achieve high accuracy and is not competitive with the BF baseline or SHL. On the other hand, ES is quite competitive with even DHL and AHL. ES retrieves the top 1000 triangles on the AMiner and MAG datasets with 99% accuracy at speedups of 2x or 4x over DHL and AHL. A similar speedup is observed for $k = 100,000$, but ES is only achieving 50% accuracy in these cases. Even though ES works well in these cases, our deterministic algorithms are still competitive; we conclude that intelligent deterministic approaches work extremely well for finding top weighted triangles in large weighted graphs.

3. All of our algorithms except BF and WS use a pre-processing step of sorting edges by weight. Surprisingly, we find that this pre-processing step is the bottleneck in our computations. Sorting in parallel is crucial to achieving high performance. In turn, this negates the possible benefit of parallel sampling for the randomized algorithms over our deterministic methods, whose main routines are inherently sequential.

Table 5.2: Running times (in seconds) averaged over 10 runs for our proposed algorithms and baselines. We ran ES to achieve 94% (k = 1,000) or 50% (k = 100,000) accuracy. We ran WS to achieve just 50% accuracy and we stopped it early if it took longer than BF (or longer than SHL on the Spotify dataset). SHL is also an approximation, so we report its accuracy in the final column.

k	dataset	BF	ES	WS	DHL	AHL	SHL	Accuracy (SHL)
	tags-so	12.71	0.57	11.28	**.08**	0.09	0.54	0.99
1000	threads-so	34.92	1.31	>BF	0.53	**0.38**	1.55	0.55
	Wikipedia	16.32	14.31	>BF	**5.44**	7.26	2.02	0.87
	Ethereum	52.91	9.03	>BF	8.12	**6.94**	11.90	0.28
	Aminer	243.75	**3.72**	>BF	13.35	12.36	43.47	0.32
	reddit-reply	4047.62	5.19	341.17	5.02	**4.74**	102.65	0.51
	MAG	512.24	**4.92**	48.58	29.19	20.89	72.49	0.91
	Spotify	>86400	60.33	>SHL	31.82	**30.79**	5388.45	1.00
	tags-so	13.06	0.58	>BF	**0.23**	**0.23**	0.62	0.28
100000	threads-so	33.99	**1.19**	>BF	1.82	1.73	1.63	0.32
	Wikipedia	17.34	13.64	>BF	**5.49**	7.24	2.15	0.13
	Ethereum	57.35	10.03	>BF	**18.11**	19.87	11.70	0.11
	Aminer	245.28	3.45	>BF	15.38	**13.91**	43.28	0.24
	reddit-reply	3857.57	**6.52**	>BF	6.87	7.49	98.34	0.34
	MAG	524.80	**4.25**	>BF	29.52	21.37	75.97	0.10
	Spotify	>86400	47.27	>SHL	30.57	**29.89**	5384.17	0.92

5.6 Additional Numerical Experiments

While our algorithms focus on finding top weighted triangles, some of the randomized sampling algorithms naturally extend to finding larger cliques. In this case the weight of a clique is the generalized p-mean of the weights of the edges in the clique. We find that the extension of edge sampling (Section 5.4.5) performs best in practice. So, we compare the performance of this edge sampling

Table 5.3: Summary statistics of datasets used in the clique sampling experiments.

dataset	# nodes	# edges	edge weight	
			mean	max
email-Enron	143	1800	16	819
email-Eu	979	29K	25	48K
contact-high-school	327	5.8K	32	29K
contact-primary-school	242	8.3K	15	780
tags-math-sx	1.6K	91K	17	16K

algorithm to an intelligent brute force approach for finding top weighted 4-cliques and 5-cliques that enumerates all cliques using the algorithm of Chiba and Nishizeki [Chiba and Nishizeki, 1985]. Our main finding is that edge sampling can approximately retrieve the top weighted cliques much faster; however, the performance does show higher variance than for the case of triangles.

Datasets. Since brute force enumeration of even 4-cliques is computationally expensive, we use smaller datasets for these experiments than the ones in Table 5.1. We construct weighted graphs from 5 temporal hypergraph datasets [Benson et al., 2018], where the weight of an edge (u, v) is the number of hyperedges that contain nodes u and v. Table 5.3 shows summary statistics for the data.

Experimental setup. We evaluate the performance of our proposed algorithm, randomized edge sampling (ES) (Algorithm 8) with the modifications mentioned in Section 5.4.5. We use an optimized sequential clique enumerator as a baseline and refer to this as the “brute force approach” (BF). The rest of the experimental setup is the same as in Section 5.5. We evaluate the algorithms for two values of k $(1,000$ and $100,000)$ and two clique sizes (4 and 5). We also use $p = 1$ in Eq. (5.7).

Since ES is randomized, it is not guaranteed to enumerate all of the top-k weighted cliques. Therefore, we measure its performance in terms of the running time and accuracy (the fraction of top-k cliques actually enumerated). In particular, we measure the time ES takes to achieve at least 50% accuracy on all datasets for 4-cliques and at least 40% accuracy for 5-cliques.

Results. Table 5.4 shows the running times of BF and ES for finding top weighted 4-cliques and 5-cliques. There are a few important takeaways.

1. ES is substantially faster than BF on all datasets for both 4-cliques and 5-cliques. For example, ES is 14x faster than BF on the email-Eu dataset for both 4-cliques and 5-cliques. However, it is important to note that ES is only required to achieve 40 to 50% accuracy in these cases.
2. Unlike the performance of ES for finding top weighted triangles, the performance of ES for finding top weighted cliques has higher variance across runs. Furthermore, in some cases, ES takes longer than BF, e.g., for finding 5-cliques on the tags-math-sx dataset. A better understanding of these changes in performance compared to the triangle case is still an open question.

Table 5.4: Running times (in seconds) averaged over 10 runs for brute force enumeration (BF) and parallel edge sampling (ES) for 4 and 5 cliques. We ran ES long enough to achieve 50% (4-cliques) and 40% (5-cliques) accuracy for $k = 1,000$ and $100,000$.

		clique size			
		4		5	
k	dataset	BF	ES	BF	ES
1000	email-Enron	1.13	0.05	1.1	0.3
	email-Eu	9.17	0.4	83	5.8
	contact-high-school	1.32	0.08	1.24	0.5
	contact-primary-school	1.63	0.3	9	2.5
	tags-math-sx	398	4.5	9340	>9340
100000	email-Enron	1.13	0.05	1.1	0.3
	email-Eu	9.17	0.45	83	5.8
	contact-high-school	1.32	0.08	1.24	0.5
	contact-primary-school	1.63	0.2	9	2.5
	tags-math-sx	398	4.5	9340	>9340

5.7 Discussion

Subgraph patterns, and in particular, triangles, have been used extensively in network science applications. However, most of the existing literature focuses on counting or enumeration tasks in unweighted graphs. In this chapter we developed deterministic and randomized sampling algorithms for finding the heaviest triangles in large weighted graphs. With some tuning, our main deterministic algorithm can find these triangles in a few seconds on graphs with hundreds of millions of edges or in 30 seconds on a graph with billions of edges. This is orders of magnitude faster than what one could achieve with existing fast enumeration schemes and is usually much faster than even our randomized sampling algorithms. We anticipate that our work will enable learners to better explore large-scale weighted graphs and therefore, better understand past interaction data.

We also expect that our work will spur new algorithmic developments on subgraph listing and counting in weighted graphs. For example, one interesting avenue for future research is to develop randomized sampling algorithms that sample triangles with probability proportional to some arbitrary function of their weight, where the function is chosen to converge to the top weighted triangles faster. This could make randomized sampling approaches competitive with our fast deterministic methods.

BIBLIOGRAPHY

Yasin Abbasi-Yadkori and Csaba Szepesvári. Regret bounds for the adaptive control of linear quadratic systems. In *COLT 2011 - The 24th Annual Conference on Learning Theory, June 9-11, 2011, Budapest, Hungary*, pages 1–26, 2011.

Jacob D. Abernethy, Elad Hazan, and Alexander Rakhlin. Competing in the dark: An efficient algorithm for bandit linear optimization. In *21st Annual Conference on Learning Theory - COLT 2008, Helsinki, Finland, July 9-12, 2008*, pages 263–274, 2008.

Alekh Agarwal, Nan Jiang, Sham M Kakade, and Wen Sun. Reinforcement learning: Theory and algorithms. 2019a.

Naman Agarwal, Brian Bullins, Elad Hazan, Sham M. Kakade, and Karan Singh. Online control with adversarial disturbances. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, pages 111–119, 2019b.

Naman Agarwal, Elad Hazan, and Karan Singh. Logarithmic regret for online control. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 10175–10184, 2019c.

Akshay Agrawal, Robin Verschueren, Steven Diamond, and Stephen Boyd. A rewriting system for convex optimization problems. *Journal of Control and Decision*, 5(1):42–60, 2018.

Shipra Agrawal and Nikhil R. Devanur. Bandits with concave rewards and convex knapsacks. In *ACM Conference on Economics and Computation, EC '14, Stanford , CA, USA, June 8-12, 2014*, pages 989–1006, 2014.

Shipra Agrawal and Nikhil R. Devanur. Linear contextual bandits with knapsacks. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 3450–3458, 2016.

Nesreen K. Ahmed, Nick G. Duffield, Jennifer Neville, and Ramana Rao Kompella. Graph sample and hold: A framework for big-graph analytics. In *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*, pages 1446–1455, 2014.

Noga Alon, Raphael Yuster, and Uri Zwick. Finding and counting given length cycles. *Algorithmica*, 17(3):209–223, 1997.

Jason M. Altschuler and Kunal Talwar. Online learning over a finite action set with limited switching. In *Conference On Learning Theory, COLT 2018, Stockholm, Sweden, 6-9 July 2018*, pages 1569–1573, 2018.

Aris Anagnostopoulos, Carlos Castillo, Adriano Fazzone, Stefano Leonardi, and Evimaria Terzi. Algorithms for hiring and outsourcing in the online labor market. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*, pages 1109–1118, 2018.

Oren Anava, Elad Hazan, and Shie Mannor. Online learning for adversaries with memory: Price of past mistakes. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 784–792, 2015.

James Anderson, John C Doyle, Steven H Low, and Nikolai Matni. System level synthesis. *Annual Reviews in Control*, 47:364–393, 2019.

Shaikh Arifuzzaman, Maleq Khan, and Madhav V. Marathe. PATRIC: a parallel algorithm for counting triangles in massive networks. In *22nd ACM International Conference on Information and Knowledge Management, CIKM'13, San Francisco, CA, USA, October 27 - November 1, 2013*, pages 529–538, 2013.

Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Mach. Learn.*, 47(2-3):235–256, 2002a.

Peter Auer, Nicolò Cesa-Bianchi, Yoav Freund, and Robert E. Schapire. The non-stochastic multiarmed bandit problem. *SIAM J. Comput.*, 32(1):48–77, 2002b.

Haim Avron. Counting triangles in large graphs using randomized matrix trace estimation. In *Workshop on Large-scale Data Mining: Theory and Applications*, 2010.

Baruch Awerbuch and Robert Kleinberg. Online linear optimization and adaptive routing. *J. Comput. Syst. Sci.*, 74(1):97–114, 2008.

Moshe Babaioff, Shaddin Dughmi, Robert Kleinberg, and Aleksandrs Slivkins. Dynamic pricing with limited supply. In *Proceedings of the 13th ACM Conference* *on Electronic Commerce, EC 2012, Valencia, Spain, June 4-8, 2012*, pages 74–91, 2012.

Ashwinkumar Badanidiyuru, Robert Kleinberg, and Aleksandrs Slivkins. Bandits with knapsacks. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, pages 207–216, 2013.

Ashwinkumar Badanidiyuru, Robert Kleinberg, and Aleksandrs Slivkins. Bandits with knapsacks. *Journal of the ACM (JACM)*, 65(3):1–55, 2018.

Siddhartha Banerjee, Carlos Riquelme, and Ramesh Johari. Pricing in ride-share platforms: A queueing-theoretic approach. *Available at SSRN 2568258*, 2015.

Alain Barrat, Marc Barthelemy, Romualdo Pastor-Satorras, and Alessandro Vespignani. The architecture of complex weighted networks. *Proceedings of the national academy of sciences*, 101(11):3747–3752, 2004.

Austin R Benson, David F Gleich, and Jure Leskovec. Higher-order organization of complex networks. *Science*, 353(6295):163–166, 2016.

Austin R. Benson, Rediet Abebe, Michael T. Schaub, Ali Jadbabaie, and Jon M. Kleinberg. Simplicial closure and higher-order link prediction. *Proc. Natl. Acad. Sci. USA*, 115(48):E11221–E11230, 2018.

Jonathan W Berry, Daniel J Nordman, Cynthia A Phillips, and Alyson G Wilson. Listing triangles in expected linear time on a class of power law graphs. Technical report, Technical report, Sandia National Laboratories, 2010.

Jonathan W Berry, Bruce Hendrickson, Randall A LaViolette, and Cynthia A Phillips. Tolerating the community detection resolution limit with edge weighting. *Physical Review E*, 83(5), 2011.

Jonathan W. Berry, Luke A. Fostvedt, Daniel J. Nordman, Cynthia A. Phillips, C. Seshadhri, and Alyson G. Wilson. Why do simple algorithms for triangle enumeration work in the real world? *Internet Mathematics*, 11(6):555–571, 2015.

Omar Besbes and Assaf Zeevi. Dynamic pricing without knowing the demand function: Risk bounds and near-optimal algorithms. *Oper. Res.*, 57(6):1407–1420, 2009.

Kush Bhatia and Karthik Sridharan. Online learning with dynamics: A minimax perspective. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Boubekeur Boukhezzar and Houria Siguerdidjane. Comparison between linear and nonlinear control strategies for variable speed wind turbines. *Control Engineering Practice*, 18:1357–1368, 2010.

Marco Bressan, Flavio Chierichetti, Ravi Kumar, Stefano Leucci, and Alessandro Panconesi. Counting graphlets: Space vs time. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, WSDM 2017, Cambridge, United Kingdom, February 6-10, 2017*, pages 557–566, 2017.

Gavin Brown, Shlomi Hod, and Iden Kalemaj. Performative prediction in a stateful world. In *International Conference on Artificial Intelligence and Statistics, AISTATS 2022, 28-30 March 2022, Virtual Event*, pages 6045–6061, 2022.

Sébastien Bubeck and Nicolò Cesa-Bianchi. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Found. Trends Mach. Learn.*, 5:1–122, 2012.

Sébastien Bubeck, Ronen Eldan, and Yin Tat Lee. Kernel-based methods for bandit convex optimization. *J. ACM*, 68(4):25:1–25:35, 2021.

Ronald S Burt. Secondhand brokerage: Evidence on the importance of local structure for managers, bankers, and analysts. *Academy of Management Journal*, 50(1): 119–148, 2007.

Asaf B. Cassel and Tomer Koren. Bandit linear control. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Lin Chen, Qian Yu, Hannah Lawrence, and Amin Karbasi. Minimax regret of switching-constrained online convex optimization: No phase transition. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Norishige Chiba and Takao Nishizeki. Arboricity and subgraph listing algorithms. *SIAM J. Comput.*, 14(1):210–223, 1985.

Shumo Chu and James Cheng. Triangle listing in massive networks and its applications. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, CA, USA, August 21-24, 2011*, pages 672–680, 2011.

Noshir S Contractor, Stanley Wasserman, and Katherine Faust. Testing multitheoretical, multilevel hypotheses about organizational networks: An analytic framework and empirical example. *Academy of Management Review*, 31(3):681–703, 2006.

Thomas M Cover. Universal portfolios. *Mathematical finance*, 1(1):1–29, 1991.

Maximilien Danisch, Oana Balalau, and Mauro Sozio. Listing k-cliques in sparse real-world graphs. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web, WWW 2018, Lyon, France, April 23-27, 2018*, pages 589–598, 2018.

Sarah Dean, Horia Mania, Nikolai Matni, Benjamin Recht, and Stephen Tu. Regret bounds for robust adaptive control of the linear quadratic regulator. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 4192–4201, 2018.

Ofer Dekel, Ambuj Tewari, and Raman Arora. Online bandit learning against an adaptive adversary: from regret to policy regret. In *Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012*, 2012.

Steven Diamond and Stephen Boyd. CVXPY: A Python-embedded modeling language for convex optimization. *J. Mach. Learn. Res.*, 17(83):1–5, 2016.

Chase P. Dowling, Lillian J. Ratliff, and Baosen Zhang. Modeling curbside parking as a network of finite capacity queues. *IEEE Trans. Intell. Transp. Syst.*, 21(3): 1011–1022, 2020.

Nurcan Durak, Ali Pinar, Tamara G Kolda, and C Seshadhri. Degree relations of triangles in real-world networks and graph models. In *21st ACM International Conference on Information and Knowledge Management, CIKM'12, Maui, HI, USA, October 29 - November 02, 2012*, pages 1712–1716, 2012.

Talya Eden, Amit Levi, Dana Ron, and C. Seshadhri. Approximately counting triangles in sublinear time. *SIAM J. Comput.*, 46(5):1603–1646, 2017.

Roohollah Etemadi, Jianguo Lu, and Yung H. Tsin. Efficient estimation of triangles in very large graphs. In *Proceedings of the 25th ACM International Conference on Information and Knowledge Management, CIKM 2016, Indianapolis, IN, USA, October 24-28, 2016*, pages 1251–1260, 2016.

Eyal Even-Dar, Shie Mannor, and Yishay Mansour. PAC bounds for multi-armed bandit and markov decision processes. In *Computational Learning Theory, 15th Annual Conference on Computational Learning Theory, COLT 2002, Sydney, Australia, July 8-10, 2002, Proceedings*, pages 255–270, 2002.

Arthur Flajolet and Patrick Jaillet. Logarithmic regret bounds for bandits with knapsacks. *arXiv preprint arXiv:1510.01800v4*, 2015.

Dylan J. Foster and Max Simchowitz. Logarithmic regret for adversarial online control. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, pages 3211–3221, 2020.

Dylan J. Foster, Alexander Rakhlin, Ayush Sekhari, and Karthik Sridharan. On the complexity of adversarial decision making. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

Xiand Gao, Xiaobo Li, and Shuzhong Zhang. Online learning with non-convex losses and non-stationary regret. In *International Conference on Artificial Intelligence and Statistics, AISTATS 2018, 9-11 April 2018, Playa Blanca, Lanzarote, Canary Islands, Spain*, pages 235–243, 2018.

David F Gleich and C Seshadhri. Vertex neighborhoods, low conductance cuts, and good seeds for local community methods. In *The 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '12, Beijing, China, August 12-16, 2012*, pages 597–605, 2012.

Gene H. Golub and Charles F. Van Loan. *Matrix Computations, Third Edition*. John Hopkins University Press, 1996.

Paula Gradu, John Hallman, and Elad Hazan. Non-stochastic control with bandit feedback. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Mohammad Al Hasan and Vachik S. Dave. Triangle counting in large networks: A review. *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.*, 8(2), 2018.

Elad Hazan. *Introduction to Online Convex Optimization*. MIT Press, 2022.

Elad Hazan and Yuanzhi Li. An optimal algorithm for bandit convex optimization. *CoRR*, abs/1603.04350, 2016.

Elad Hazan, Sham M. Kakade, and Karan Singh. The nonstochastic control problem. In *Algorithmic Learning Theory, ALT 2020, 8-11 February 2020, San Diego, CA, USA*, pages 408–421, 2020.

Keith Henderson, Brian Gallagher, Tina Eliassi-Rad, Hanghang Tong, Sugato Basu, Leman Akoglu, Danai Koutra, Christos Faloutsos, and Lei Li. RolX: Structural role extraction & mining in large graphs. In *The 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '12, Beijing, China, August 12-16, 2012*, 2012.

Jack Hessel, Chenhao Tan, and Lillian Lee. Science, askscience, and badscience: On the coexistence of highly related communities. In *Proceedings of the Tenth International Conference on Web and Social Media, Cologne, Germany, May 17-20, 2016*, pages 171–180, 2016.

John J Horton. Online labor markets. In *Internet and Network Economics - 6th International Workshop, WINE 2010, Stanford, CA, USA, December 13-17, 2010. Proceedings*, pages 515–522, 2010.

Nicole Immorlica, Karthik Abinav Sankararaman, Robert E. Schapire, and Aleksandrs Slivkins. Adversarial bandits with knapsacks. *J. ACM*, 69(6):40:1–40:47, 2022.

Meena Jagadeesan, Tijana Zrnic, and Celestine Mendler-Dünner. Regret minimization with performative feedback. In *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, 2022.

Shweta Jain and C. Seshadhri. A fast and provable method for estimating clique counts using turán's theorem. In *Proceedings of the 26th International Conference on World Wide Web, WWW 2017, Perth, Australia, April 3-7, 2017*, pages 441–449, 2017.

Madhav Jha, C. Seshadhri, and Ali Pinar. Path sampling: A fast and provable method for estimating 4-vertex subgraph counts. In *WWW*, pages 495–505, 2015.

Junteng Jia, Michael T. Schaub, Santiago Segarra, and Austin R. Benson. Graph-based semi-supervised & active learning for edge flows. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019*, pages 761–771, 2019.

Anna Karlin. Lecture notes for cse 522: Algorithms and uncertainty, 2017. URL https://courses.cs.washington.edu/courses/cse522/17sp/lectures/Lecture9.pdf.

Robert Kleinberg, Aleksandrs Slivkins, and Eli Upfal. Bandits and experts in metric spaces. *J. ACM*, 66(4):30:1–30:77, 2019.

Robert David Kleinberg. *Online Decision Problems with Large Strategy Sets*. PhD thesis, Massachusetts Institute of Technology, 2005.

Tamara G. Kolda, Ali Pinar, and C. Seshadhri. Triadic measures on graphs: The power of wedge sampling. In *Proceedings of the 13th SIAM International Conference on Data Mining, May 2-4, 2013. Austin, Texas, USA*, pages 10–18, 2013.

Mihail N Kolountzakis, Gary L Miller, Richard Peng, and Charalampos E Tsourakakis. Efficient triangle counting in large graphs via degree-based vertex partitioning. *Internet Mathematics*, 8(1-2):161–185, 2012.

Vladimír Kučera. Stability of discrete linear feedback systems. *IFAC Proceedings Volumes*, 8(1):573–578, 1975.

Raunak Kumar and Robert Kleinberg. Non-monotonic resource utilization in the bandits with knapsacks problem. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022a.

Raunak Kumar and Robert Kleinberg. Non-monotonic resource utilization in the bandits with knapsacks problem, 2022b. URL https://github.com/raunakkmr/non-monotonic-resource-utilization-in-the-bandits-with-knapsacks-proble

Raunak Kumar, Paul Liu, Moses Charikar, and Austin Benson. Retrieving top weighted triangles in graphs. In *WSDM '20: The Thirteenth ACM International*

Conference on Web Search and Data Mining, Houston, TX, USA, February 3-7, 2020, pages 295–303, 2020.

Raunak Kumar, Sarah Dean, and Robert Kleinberg. Online convex optimization with unbounded memory. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

Tze Leung Lai and Herbert Robbins. Asymptotically efficient adaptive allocation rules. *Advances in applied mathematics*, 6(1):4–22, 1985.

Peter Lancaster and Leiba Rodman. *Algebraic Riccati Equations*. Clarendon press, 1995.

Matthieu Latapy. Main-memory triangle computations for very large (sparse (power-law)) graphs. *Theor. Comput. Sci.*, 407(1-3):458–473, 2008.

Tor Lattimore and Csaba Szepesvári. *Bandit algorithms*. Cambridge University Press, 2020.

Barbara S Lawrence. Organizational reference groups: A missing perspective on social context. *Organization Science*, 17(1):80–100, 2006.

Nevena Lazic, Craig Boutilier, Tyler Lu, Eehern Wong, Binz Roy, M. K. Ryu, and Greg Imwalle. Data center cooling using model-predictive control. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 3818–3827, 2018.

Xiaocheng Li, Chunlin Sun, and Yinyu Ye. The symmetry between arms and knapsacks: A primal-dual approach for bandits with knapsacks. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, pages 6483–6492, 2021a.

Yingying Li, Subhro Das, and Na Li. Online optimal control with affine constraints. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 8527–8537, 2021b.

Yiheng Lin, James Preiss, Emile Anand, Yingying Li, Yisong Yue, and Adam Wierman. Online adaptive policy selection in time-varying systems: No-regret via contractive perturbations. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

Nick Littlestone and Manfred K. Warmuth. The weighted majority algorithm. In *30th Annual Symposium on Foundations of Computer Science, Research Triangle Park, North Carolina, USA, 30 October - 1 November 1989*, pages 256–261. IEEE Computer Society, 1989.

Nick Littlestone and Manfred K. Warmuth. The weighted majority algorithm. *Inf. Comput.*, 108(2):212–261, 1994.

Paul Liu, Austin R. Benson, and Moses Charikar. Sampling methods for counting temporal motifs. In *WSDM*, pages 294–302, 2019.

Kristian Lum and William Isaac. To predict and serve? *Significance*, 13(5):14–19, 2016.

Priya Mahadevan, Calvin Hubble, Dmitri Krioukov, Bradley Huffaker, and Amin Vahdat. Orbis: Rescaling degree correlations to generate annotated internet topologies. In *Proceedings of the ACM SIGCOMM 2007 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, Kyoto, Japan, August 27-31, 2007*, pages 325–336, 2007.

Celestine Mendler-Dünner, Juan C. Perdomo, Tijana Zrnic, and Moritz Hardt. Stochastic optimization for performative prediction. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

John Miller, Juan C. Perdomo, and Tijana Zrnic. Outside the echo chamber: Optimizing the performative risk. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, pages 7710–7720, 2021.

Ron Milo, Shai Shen-Orr, Shalev Itzkovitz, Nadav Kashtan, Dmitri Chklovskii, and Uri Alon. Network motifs: Simple building blocks of complex networks. *Science*, 298(5594):824–827, 2002.

Ron Milo, Shalev Itzkovitz, Nadav Kashtan, Reuven Levitt, Shai Shen-Orr, Inbal Ayzenshtat, Michal Sheffer, and Uri Alon. Superfamilies of evolved and designed networks. *Science*, 2004.

Edgar Minasyan, Paula Gradu, Max Simchowitz, and Elad Hazan. Online control of unknown time-varying dynamical systems. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, 2021.

Mark EJ Newman, Steven H Strogatz, and Duncan J Watts. Random graphs with arbitrary degree distributions and their applications. *Physical Review E*, 64(2), 2001.

Jukka-Pekka Onnela, Jari Saramäki, János Kertész, and Kimmo Kaski. Characterizing motifs in weighted complex networks. In *AIP Conference Proceedings*, 2005a.

Jukka-Pekka Onnela, Jari Saramäki, János Kertész, and Kimmo Kaski. Intensity and coherence of motifs in weighted complex networks. *Phys. Rev. E*, 71, 2005b.

Tore Opsahl and Pietro Panzarasa. Clustering in weighted networks. *Social Networks*, 31(2):155–163, 2009.

Francesco Orabona. A modern introduction to online learning. *CoRR*, abs/1912.13213, 2019.

Rasmus Pagh and Charalampos E Tsourakakis. Colorful triangle counting and a mapreduce implementation. *Information Processing Letters*, 112(7):277–281, 2012.

Juan C. Perdomo, Tijana Zrnic, Celestine Mendler-Dünner, and Moritz Hardt. Performative prediction. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, pages 7599–7609, 2020.

Gregory Pierce and Donald Shoup. Sfpark: Pricing parking by demand. In *Parking and the City*, pages 344–353. Routledge, 2018.

Nataša Pržulj. Biological network comparison using graphlet degree distribution. *Bioinformatics*, 23(2):e177–e183, 2007.

Yuzhen Qin, Yingcong Li, Fabio Pasqualetti, Maryam Fazel, and Samet Oymak. Stochastic contextual bandits with long horizon rewards. In *Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI 2023, Washington, DC, USA, February 7-14, 2023*, 2023.

Mahmudur Rahman and Mohammad Al Hasan. Approximate triangle counting algorithms on multi-cores. In *2013 IEEE International Conference on Big Data (IEEE BigData 2013), 6-9 October 2013, Santa Clara, CA, USA*, pages 127–133, 2013.

Mahmudur Rahman and Mohammad Al Hasan. Sampling triples from restricted networks using MCMC strategy. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, CIKM 2014, Shanghai, China, November 3-7, 2014*, pages 1519–1528, 2014.

Mitas Ray, Lillian J. Ratliff, Dmitriy Drusvyatskiy, and Maryam Fazel. Decision-dependent risk minimization in geometrically decaying dynamic environments. In *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 - March 1, 2022*, pages 8081–8088, 2022.

Herbert Robbins. Some aspects of the sequential design of experiments. *Bulletin of the American Mathematical Society*, 58:527–535, 1951.

Garry Robins, Pip Pattison, Yuval Kalish, and Dean Lusher. An introduction to exponential random graph p^* models for social networks. *Social Networks*, 29 (2):173–191, 2007.

Pablo Robles-Granda, Sebastián Moreno, and Jennifer Neville. Sampling of attributed networks from hierarchical generative models. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 1155–1164, 2016.

Karl Rohe and Tai Qin. The blessing of transitivity in sparse and stochastic networks. *arXiv preprint arXiv:1307.2302*, 2013.

Ryan A. Rossi and Nesreen K. Ahmed. Role discovery in networks. *IEEE Transactions on Knowledge and Data Engineering*, 27(4):1112–1131, 2015.

Karthik Abinav Sankararaman and Aleksandrs Slivkins. Bandits with knapsacks beyond the worst case. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 23191–23204, 2021.

Thomas Schank and Dorothea Wagner. Approximating clustering coefficient and transitivity. *J. Graph Algorithms Appl.*, 9(2):265–275, 2005.

C. Seshadhri, Ali Pinar, and Tamara G. Kolda. Wedge sampling for computing clustering coefficients and triangle counts on large graphs. *Statistical Analysis and Data Mining*, 7(4):294–307, 2014.

Shai Shalev-Shwartz. Online learning and online convex optimization. *Found. Trends Mach. Learn.*, 4(2):107–194, 2012.

Shai Shalev-Shwartz and Yoram Singer. Online learning meets optimization in the dual. In *Learning Theory, 19th Annual Conference on Learning Theory, COLT 2006, Pittsburgh, PA, USA, June 22-25, 2006, Proceedings*, 2006.

Guanya Shi, Yiheng Lin, Soon-Jo Chung, Yisong Yue, and Adam Wierman. Online optimization with memory and competitive control. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Max Simchowitz and Dylan J. Foster. Naive exploration is optimal for online LQR. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, 2020.

Arnab Sinha, Zhihong Shen, Yang Song, Hao Ma, Darrin Eide, Bo-June Paul Hsu, and Kuansan Wang. An overview of microsoft academic service (MAS) and applications. In *Proceedings of the 24th International Conference on World Wide Web Companion, WWW 2015, Florence, Italy, May 18-22, 2015 - Companion Volume*, pages 243–246, 2015.

Aleksandrs Slivkins. Contextual bandits with similarity information. In *COLT*

2011 - The 24th Annual Conference on Learning Theory, June 9-11, 2011, Budapest, Hungary, pages 679–702, 2011.

Aleksandrs Slivkins. Contextual bandits with similarity information. *J. Mach. Learn. Res.*, 15(1):2533–2568, 2014.

Aleksandrs Slivkins. Introduction to multi-armed bandits. *Found. Trends Mach. Learn.*, 12:1–286, 2019.

Hossein Azari Soufiani and Edoardo M. Airoldi. Graphlet decomposition of a weighted network. In *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2012, La Palma, Canary Islands, Spain, April 21-23, 2012*, pages 54–63, 2012.

Lorenzo De Stefani, Alessandro Epasto, Matteo Riondato, and Eli Upfal. Trièst: Counting local and global triangles in fully dynamic streams with fixed memory size. *ACM Trans. Knowl. Discov. Data*, 11(4), 2017a.

Lorenzo De Stefani, Erisa Terolli, and Eli Upfal. Tiered sampling: An efficient method for approximate counting sparse motifs in massive graph streams. In *2017 IEEE International Conference on Big Data (IEEE BigData 2017), Boston, MA, USA, December 11-14, 2017*, pages 776–786, 2017b.

Arun Sai Suggala and Praneeth Netrapalli. Online non-convex learning: Following the perturbed leader is optimal. In *Algorithmic Learning Theory, ALT 2020, 8-11 February 2020, San Diego, CA, USA*, pages 845–861, 2020.

Siddharth Suri and Sergei Vassilvitskii. Counting triangles and the curse of the last reducer. In *Proceedings of the 20th International Conference on World Wide Web, WWW 2011, Hyderabad, India, March 28 - April 1, 2011*, pages 607–614, 2011.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An Introduction*. MIT press, 2018.

Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. Arnetminer: Extraction and mining of academic social networks. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Las Vegas, Nevada, USA, August 24-27, 2008*, pages 990–998, 2008.

Long Tran-Thanh, Archie C. Chapman, Enrique Munoz de Cote, Alex Rogers, and Nicholas R. Jennings. Epsilon-first policies for budget-limited multi-armed bandits. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2010, Atlanta, Georgia, USA, July 11-15, 2010*, pages 1211–1216, 2010.

Long Tran-Thanh, Archie C. Chapman, Alex Rogers, and Nicholas R. Jennings. Knapsack based optimal policies for budget-limited multi-armed bandits. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, July 22-26, 2012, Toronto, Ontario, Canada*, pages 1134–1140, 2012.

Charalampos E. Tsourakakis. Fast counting of triangles in large real networks without counting: Algorithms and laws. In *Proceedings of the 8th IEEE International Conference on Data Mining (ICDM 2008), December 15-19, 2008, Pisa, Italy*, pages 608–617. IEEE Computer Society, 2008.

Charalampos E Tsourakakis, U Kang, Gary L Miller, and Christos Faloutsos. Doulion: Counting triangles in massive graphs with a coin. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Paris, France, June 28 - July 1, 2009*, pages 837–846, 2009.

Charalampos E Tsourakakis, Mihail N Kolountzakis, and Gary L Miller. Triangle sparsifiers. *J. Graph Algorithms Appl.*, 15(6):703–726, 2011.

Duru Türkoglu and Ata Turk. Edge-based wedge sampling to estimate triangle counts in very large graphs. In *2017 IEEE ICDM 2017*, pages 455–464, 2017.

Martin J Wainwright. *High-dimensional Statistics: A Non-asymptotic Viewpoint*. Cambridge University Press, 2019.

Yuh-Shyang Wang, Nikolai Matni, and John C. Doyle. A system-level approach to controller synthesis. *IEEE Trans. Autom. Control.*, 64(10):4079–4093, 2019.

Stanley Wasserman and Katherine Faust. *Social Network Analysis: Methods and Applications*. Cambridge University Press, 1994.

Brooke Foucault Welles, Anne Van Devender, and Noshir Contractor. Is a friend a friend?: Investigating the structure of friendship networks in virtual worlds. In *CHI '10 Extended Abstracts on Human Factors in Computing Systems*, pages 4027–4032, 2010.

Thomas H Wolff. *Lectures on Harmonic Analysis*. American Mathematical Society, 2003.

Ellery Wulczyn and Dario Taraborelli. Wikipedia clickstream, 2017.

Kuai Xu, Feng Wang, and Lin Gu. Behavior analysis of internet traffic via bipartite graphs and one-mode projections. *IEEE/ACM Trans. Netw.*, 22(3):931–942, 2014.

Hao Yin, Austin R. Benson, and Jure Leskovec. The local closure coefficient: A new perspective on network clustering. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining, WSDM 2019, Melbourne, VIC, Australia, February 11-15, 2019*, pages 303–311, 2019.

Dante Youla, Hamid Jabr, and Jr Bongiorno. Modern wiener-hopf design of optimal controllers–part ii: The multivariable case. *IEEE Transactions on Automatic Control*, 21(3):319–338, 1976.

Lijun Zhang, Tianbao Yang, Rong Jin, and Zhi-Hua Zhou. Online bandit learning for a special class of non-convex losses. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA*, pages 3158–3164, 2015.

Peng Zhao, Guanghui Wang, Lijun Zhang, and Zhi-Hua Zhou. Bandit convex optimization in non-stationary environments. *J. Mach. Learn. Res.*, 22:125:1–125:45, 2021.

Peng Zhao, Yu-Xiang Wang, and Zhi-Hua Zhou. Non-stationary online learning with memory and non-stochastic control. In *International Conference on Artificial Intelligence and Statistics, AISTATS 2022, 28-30 March 2022, Virtual Event*, pages 2101–2133, 2022.

Martin Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the 20th International Conference on Machine Learning, ICML 2003, August 21-24 2003, Washington, DC, USA*, 2003.