1-1-2015

New Models qnd Algorithms for Bandits and Markets

Kareem Amin
University of Pennsylvania, akareemamin@gmail.com

New Models qnd Algorithms for Bandits and Markets

Abstract
Inspired by advertising markets, we consider large-scale sequential decision making problems in which a learner must deploy an algorithm to behave optimally under uncertainty. Although many of these problems can be modeled as contextual bandit problems, we argue that the tools and techniques for analyzing bandit problems with large numbers of actions and contexts can be greatly expanded. While convexity and metric-similarity assumptions on the process generating rewards have yielded some algorithms in existing literature, certain types of assumptions that have been fruitful in offline supervised learning settings have yet to even be considered. Notably missing, for example, is any kind of graphical model approach to assuming structured rewards, despite the success such assumptions have

achieved in inducing scalable learning and inference with high-dimensional distributions. Similarly, we observe that there are countless tools for understanding the relationship between a choice of model class in supervised learning, and the generalization error of the best fit from that class, such as the celebrated VC-theory. However, an analogous notion of dimensionality, which relates a generic structural assumption on rewards to regret rates in an online optimization problem, is not fully developed. The primary goal of this dissertation, therefore, will be to fill out the space of models, algorithms, and assumptions used in sequential decision making problems. Toward this end, we will develop a theory for bandit problems with structured rewards that permit a graphical model representation. We will give an efficient algorithm for regret-minimization in such a setting, and along the way will develop a deeper connection between online supervised learning and regret-minimization. This dissertation will also introduce a complexity measure for generic structural assumptions on reward functions, which we call the Haystack Dimension. We will prove that the Haystack Dimension characterizes the optimal rates achievable up to log factors. Finally, we will describe more application-oriented techniques for solving problems in advertising markets, which again demonstrate how methods from traditional disciplines, such as statistical survival analysis, can be leveraged to design novel algorithms for optimization in markets.

Degree Type
Dissertation

Degree Name
Doctor of Philosophy (PhD)

Graduate Group
Computer and Information Science

First Advisor
Michael Kearns

Keywords
Advertising, Algorithms, Bandits, Machine Learning, Markets, Regret-Minimization

NEW MODELS AND ALGORITHMS FOR BANDITS AND MARKETS

Kareem Amin

A DISSERTATION

in

Computer and Information Science

Presented to the Faculties of the University of Pennsylvania

in

Partial Fulfillment of the Requirements for the

Degree of Doctor of Philosophy

2015

Supervisor of Dissertation

Michael Kearns, Professor, Computer and Information Sciences

Graduate Group Chairperson

Lyle Ungar, Professor, Computer and Information Sciences

Dissertation Committee

Sanjeev Khanna, Professor, Computer and Information Sciences (Chair)

Alexander Rakhlin, Assistant Professor, Department of Statistics

Aaron Roth, Assistant Professor, Computer and Information Science

Jacob Abernethy, Assistant Professor, University of Michigan (External)

NEW MODELS AND ALGORITHMS FOR BANDITS AND MARKETS

Dedicated to Mom & Dad

ACKNOWLEDGEMENT

A dissertation is a reflection on many years of academic work. However, in completing mine, I cannot help but also reflect on the end of a significant portion of my personal life. While the words of this document will concern algorithms and proofs, the document itself represents a certain place and time in my life, as well as the people and relationships that inhabited that space. I owe those people acknowledgment not just for the ways in which they aided the completion of this thesis, but for the ways in which they made my early adulthood the exciting, challenging, efflorescent, and ultimately rewarding time that it was.

One night, while having dinner with a friend from a different department, he mentioned that he had attended a talk given by Michael Kearns. He mused that it was interesting to see the influence of an advisor on his student, that it was like "watching you up there." I laughed, rolled my eyes, and emptied my wine glass. However, his point stuck with me. I might even admit that hearing it made me proud.

A good advisor shapes not just one's taste in research, but all the skills necessary to develop a career. These skills, in turn, become part of one's character. Michael taught me to strive for a certain ideal: to communicate clearly and effectively, to see the bigger picture, to identify compelling questions, to take pride in my endeavors, and to reflect that pride in my work. While striving for any ideal will always be a work in progress, identifying the target is the hardest part. These lessons have not just influenced the researcher that I am today, but also the person that I am today. And for that I thank you Michael.

Among my collaborators, I must thank Umar Syed in particular. I had the amazing fortune of working with Umar when I was an uncertain fledgling researcher. His humility, selflessness, patience, and good-nature, (not to mention intelligence) represents another ideal, both personally and professionally, which I can only hope to match someday. Without his encouragement, I am not sure if I would be sitting here, completing my thesis.

I thank Jacob Abernethy not just for his academic collaborations, but also for his friendship. Jake's kindness, generosity, and openness has meant a lot to me, especially since leaving Penn. It is no easy task to move hundreds of miles from home and not to feel the impact socially. Yet somehow, in no small part because of Jake, this has happened. I am lucky to have him watching out for me.

I am naturally indebted to all my other wonderful collaborators. In no particular order, I thank Afshin Rostamizadeh, Moez Draief, Satyen Kale, Gerry Tesauro, Peter Key, Hoda Heidari, Aaron Roth, Lili Dworkin, Rachel Cummings, Shahin Jabbari, and Gagan Goel, for their ideas, time, and hard work. I also thank Sanjeev Khanna, and Shasha Rakhlin for sitting on my committee.

I thank the friends who supported me on this journey. Alex, I am so incredibly lucky to have you as a best friend. Most relationships are conditional, subject to the whims of time and circumstance. It gives me a profound psychological strength to know that I can never be truly alone. Should everything else crumble around me, I know that I will always have our friendship. Thank you for being the brother I never had.

My girls, Aimee, Emily, and Jessie, thank you for obvious reasons. Shirking work to go on some road trip, hanging out in Austin, tea with old ladies, weekends at the mansion, bean-bag toss, weird co-op parties, jumping in frozen Brooklyn water, nature hikes, marathons, setting up tent on the side of the road, all probably did not help me get my degree any quicker. But you kept me sane through the years, and helped me through some darker times as well.

Dogan, Anna, and Annie, (Ana too) thank you for giving a me a family when I least expected it. I don't think I was ever happier than when we were all together. Julia, I could always count on you to be there to climb a thing, or to discover sordid affairs. Eli, I will miss our philosophical discussions, and your solid aesthetic taste. I am still terrible at basketball. Thank you Craig for keeping me sane that one summer, and for being down to hangout

whenever we cross paths. Natalia and Selman, ΣKN 4life. I will take this opportunity to have it officially published that Tina is better than Lyla. Thank you to Lori and Eran for building a community, and creating a place where I could forget my worries at least for a little while. Thank you Viky for all the work you selflessly did, and Dinesh and Ana for carrying on the torch. Thank you to the Happy Hour in One Hour gang – Lili, Shahin, Ryan (Daniela too), Justin, Steven, Rachel, and Hoda – we had some great times together. Jenny Gillenwater, Alex Kulesza, and David Weiss, you stand out as some of my favorite people (formerly) in the department. Thank you to Chris and Ari (and all the other guys, esp., Jackie), for letting some punk kid work with you for so long. I thank Olga, Jon, and Tyler for being my original collaborators, and, of course, Dan Huttenlocher for the lake parties.

I thank Marie, who with a single tear in the Alaskan wilderness, changed something inside me for the better. I thank John, Terri and Jack for letting me into their home, and always making me feel warm and welcome. Tom, Phyllis, Devin, Sara and Jennyfer, you will always be a second family to me. Thank you to Sabry for letting me win at chess, and Terri for letting me win at Scrabble. I'm also grateful to the Grafs, for providing me with a home overseas. Jasmin, in particular, thank you for always being such a careful and gracious host. I hope I can one day return the favor. Walter, Olie, Philip, Alessandra, Nadja, and little Collien, thank you for filling those trips with so much life.

Finally, my family, I could not have done this without you. It is interesting that the deepest gratitude is somehow the hardest to express with words. I suppose that may be because no words will suffice. Dinah, Mom, and Dad, thank you for always being there for me.
I love you.

ABSTRACT

NEW MODELS AND ALGORITHMS FOR BANDITS AND MARKETS

Kareem Amin

Michael Kearns

Inspired by advertising markets, we consider large-scale sequential decision making problems in which a learner must deploy an algorithm to behave optimally under uncertainty. Although many of these problems can be modeled as contextual bandit problems, we argue that the tools and techniques for analyzing bandit problems with large numbers of actions and contexts can be greatly expanded. While convexity and metric-similarity assumptions on the process generating rewards have yielded some algorithms in existing literature, certain types of assumptions that have been fruitful in offline supervised learning settings have yet to even be considered. Notably missing, for example, is any kind of graphical model approach to assuming structured rewards, despite the success such assumptions have achieved in inducing scalable learning and inference with high-dimensional distributions. Similarly, we observe that there are countless tools for understanding the relationship between a choice of model class in supervised learning, and the generalization error of the best fit from that class, such as the celebrated VC-theory. However, an analogous notion of dimensionality, which relates a generic structural assumption on rewards to regret rates in an online optimization problem, is not fully developed. The primary goal of this dissertation, therefore, will be to fill out the space of models, algorithms, and assumptions used in sequential decision making problems. Toward this end, we will develop a theory for bandit problems with structured rewards that permit a graphical model representation. We will give an efficient algorithm for regret-minimization in such a setting, and along the way will develop a deeper connection between online supervised learning and regret-minimization. This dissertation will also introduce a complexity measure for generic structural assumptions on reward functions, which we call the Haystack Dimension. We will prove that the Haystack Dimension

characterizes the optimal rates achievable up to log factors. Finally, we will describe more application-oriented techniques for solving problems in advertising markets, which again demonstrate how methods from traditional disciplines, such as statistical survival analysis, can be leveraged to design novel algorithms for optimization in markets.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF ILLUSTRATIONS

CHAPTER 1 : Introduction

Internet advertising is a multibillion dollar industry, and often provides the most significant source of revenue to companies providing other services on the Internet. Advertisers (seeking to buy impressions), and advertising networks (seeking to sell impressions) often face the task of solving large-scale strategic optimization problems with little human intervention.

Advertisers, for example, might participate in a sequence of repeated auctions in which they bid for impressions, or opportunities to display their ad. By participating in these auctions, an advertiser learns something about the market, or the preferences of other competing parties. At the same time, the advertiser's primary goal is to win impressions that are valuable to it. How should the advertiser bid?

On the other side of the transaction, the advertising network must manage a large database of advertisements, matching them to a continuously arriving inventory of impressions. After this matching, the advertising network witnesses a reward, most often in the form of monetary revenue. How can the advertising network learn to efficiently allocate its inventory and set prices for various impressions?

Inspired by such applications, this dissertation will describe new theoretical models and techniques that address the goals of agents in these settings. The environments we consider will always feature a decision-making agent whose goal is to optimize some objective reward across time. Many of the results we describe will relate directly to the problem of online learning from limited feedback (often times called "bandit" information). While such settings have had a long history of study, we will argue that many techniques that have been greatly successful in more traditional offline or batch (usually supervised) learning settings, have yet to even be considered in the case of such sequential decision making problems.

The primary goal of this dissertation, therefore, will be to fill out the space of models, algorithms, and assumptions used in online learning settings. We will be particularly interested

in developing the online analogs of techniques which have been fruitful in offline settings.

Consider for example the ad-selection problem – deciding which ads to show to which users – faced by an advertising network. As users arrive, the advertising network must sequentially and irrevocably pick advertisements to display. Moreover, upon making a selection, the advertising network observes only the reward associated with its selection, and does not observe counterfactual information such as what reward would have been realized had a different ad been selected. Immediately, one recognizes this as a good fit for study as a mutli-armed bandit problem.

Realistically, however, the learner in this case is faced with a large set of possible actions (a database of millions of ads), and a rich set of high-dimensional features that may be predictive of the user's behavior (describing, for example, the webpage being viewed by the user). Thus, in tackling this problem, our desiderata must include scalability in both the number of actions and the size of the aforementioned feature space (often called the context space).

Much of the focus of machine learning is the development of various techniques to address the so-called "cure of dimensionality." Thus, we need not look far for inspiration from offline methods. Indeed, as in much of machine learning, making a convexity assumption — that rewards are convex functions of the selected actions (or well-approximated by such a function) — yields efficient algorithms that are some version of gradient descent (Flaxman et al., 2005; Dani et al., 2007; Abernethy et al., 2008). A separate line of literature assumes that actions are embedded in some metric space, with nearby actions generating similar rewards through a Lipschitz payoff function (Agrawal, 1995; Kleinberg, 2004; Kleinberg et al., 2008; Lu et al., 2010; Slivkins, 2014). This is conceptually akin to the implicit assumptions that make nearest-neighbor approaches appealing supervised methods. In each of these cases, one can view the development of online methods as mirroring the types of assumptions that have been leveraged for decades to derive supervised learning algorithms.

Convexity and metric similarity assumptions are by far the most common tools that allow tractable solutions to large-scale bandit problems, and as previously discussed, are satisfying in that they mirror standard offline assumptions. However, we now note that conspicuously missing from the bandit literature is any kind of graphical model assumption on the structure of payoffs.

Probabilistic graphical models are powerful tools allowing conditional independence assumptions about a distribution over d variables to be succinctly described, thereby reducing the number of parameters required to fully specify such a distribution. Similarly, we might seek a succinct specification of structural assumptions on how payoffs in a multi-armed bandit problem are generated. Despite the success probabilistic graphical models have enjoyed in inducing scalable inference and estimation with high-dimensional distributions, such an assumption has notably not been transferred to the bandit literature.

Developing such a theory will be the focus of Chapters 2 and 3. In Chapter 2 we consider a setting where the multi-armed bandit algorithm knows the coarse underlying graphical or dependency structure of the payoff model, but not the actual parameters. Once again, this is analogous to knowing the conditional independence assumptions specified by a probabilistic graphical model, but not the parameters which ultimately specify distribution.

Along the way, we will see that the methods described in Chapter 2 in fact generalize, providing a deeper connection between supervised learning and bandit algorithms. In Chapter 3 we study this generalization, establishing a link between a specific supervised learning model, KWIK learning (Li et al., 2011; Strehl and Littman, 2007; Walsh et al., 2009; Sayedi et al., 2010; Li and Littman, 2010), and stochastic bandit problems in general. Specifically, we will find that parametric assumptions that yield KWIK learning algorithms with $1/\epsilon$ and $1/\epsilon^2$ rates imply efficient bandit algorithms with $\sqrt{T}$ and $T^{2/3}$ regret rates, respectively, when the same parametric assumptions hold in the bandit setting. Thus one route toward the discovery of efficient bandit algorithms is to solve the corresponding supervised learning problem. These results, furthermore, encourage the discovery of KWIK algorithms, as they

immediately yield new bandit algorithms, beyond being interesting in their own right.

In the preceding discussion we see a certain pattern unfolding. One starts with a set of assumptions regarding the payoff model. In previous work, payoffs might be Lipschitz or convex. In our case, we assume payoffs decompose according to a known graphical model representation, or belong to a parametric class of functions which admits a solution via a particular kind of supervised learning algorithm. From these assumptions, we can derive tractable algorithms. A natural question to ask is therefore: more generally, what types of assumptions suffice?

In supervised learning problems, one often begins by assuming a class of models, then asking how well such models are able to generalize from data. Statistical learning theory provides analytical techniques to address this very question, including the celebrated VC theory (Vapnik and Chervonenkis, 1971). For classification, the VC-dimension of a particular class of models allows one to relate the test error for using that class to a purely combinatorial property of the class – the largest set of points that can be *shattered* using a member classifier.

Continuing with the theme of this dissertation, we seek the corresponding analytical tool for sequential decision making problems. Namely, we would like a notion of complexity which relates the regret rate that an optimal algorithm can achieve against a class of reward models to combinatorial properties of the model class. The resulting notion, which we call the Haystack Dimension (Chapter 4), depends on identifying a finite subset of reward functions which are both difficult to identify and to maximize. Moreover, we show that the rates specified by the Haystack Dimension are tight, therefore making the Haystack Dimension the correct notion of complexity for stochastic bandit problems. More generally, the Haystack Dimension can be applied to any setting where a decision maker must balance gaining new information about the environment with attempts to succeed at its task.

In studying the bandit problem generally, the methods we describe are not specific to ad-

selection, and can be applied to other settings such as clinical trials, or financial trading. In the remaining chapters we switch our focus to results which are oriented to our initial motivating application. Nevertheless, the goal will be the same: we seek to expand the type of models used in decision making problems by drawing inspiration from classical methods.

For example, in Chapter 6, we study the problem faced by a strategic bidder in a repeated second price auction with budget constraints. Such a model is inspired by keyword auctions, where advertisers specify a budget to spend over a particular time period. We note that the advertiser receives *censored observations* of the highest competing bid. If the advertiser fails to win the impression, it receives some information, namely that the competing bid was at least as large as the submitted bid, but does not observe the competing bid directly. Such censored observations are a central feature of statistical survival analysis. When estimating the mortality of a given population, for example, surviving members of the population provide censored observations: an individual's time of death is at least as large as their current age. Utilizing statistical estimation techniques for survival analysis, namely the Kaplan-Meier estimator for censored data, we derive an algorithm with state-of-the-art empirical performance on real large-scale keyword auction data. We note that censored observations are a key feature of other markets (Ganchev et al., 2010), and thus importing such estimation techniques to sequential decision making problems is a rich area for continuing research.

In each of the results we have described, whether we are introducing a class of assumptions in the form of graphical models, or introducing an analytic characterization of complexity akin the the VC-dimension, we illustrate the value of enriching the space of techniques applied to sequential decision making problems. We now proceed to describe each of these contributions in greater detail.

1.1. Chapter 2: Graphical Bandits and the Ad-Selection Problem

Returning to our motivating application, consider the optimization problem faced by a typical advertising network. At a high level, the advertising network receives a stream of impressions, and in an online manner, must take some action for each new impression. After selecting an action, it receives a reward. This reward could be monetary, but could also correspond to optimizing click-through, pageviews, session-length, or some other performance metric. One can think of actions as corresponding to a selection of advertisement. However, in the case where an auction mechanism ultimately decides which specific ad is shown, the choice of action might correspond to what types of advertisers participate in the auction.

As previously discussed, one natural way to model the problem faced by such a decision-maker is by casting it as a multi-armed bandit problem with stochastic rewards. By assuming stochastic rewards, we implicitly take a macroscopic viewpoint on the market, and assume that our choice of algorithm will not affect advertiser behavior. That is, we do not attempt to model buyers as strategic agents who may react to our choice of strategy, as will later consider in Chapter 5.

Multi-Armed bandits have a long history of study (Robbins, 1985; Lai and Robbins, 1985; Auer et al., 2002, 2003; Audibert et al., 2010; Bubeck and Cesa-Bianchi, 2012) and are sequential decision-making problems that capture the tension in learning between exploration (gaining new information about the environment) and exploitation (leveraging that information to behave optimally). We are more specifically interested in contextual bandit problems (Langford and Zhang, 2007), which incorporates state information, called context, during each round of decision-making. On each round, the learner is presented with context, and after receiving this information takes some action. The learner then receives feedback, some measure of loss or reward, characterizing the quality of its choice. The bandit setting is characterized by the fact that the learner only observes feedback associated with its choice, and does not observe what would have occurred had a different action been selected. As opposed to reinforcement learning models, the contexts (which are analogous

to states in reinforcement learning) are thought to arrive exogenously of the algorithm's decision-making.

Online advertising has long been a motivating application for the study of contextual bandits (Langford and Zhang, 2007; Pandey et al., 2007; Li et al., 2010; Lu et al., 2010). Contexts correspond to features describing the impression, while actions correspond to a selection of advertisement. Nevertheless, there is a significant hurdle to applying standard bandit algorithms to this setting: the shear number of actions and contexts available to the learner. An advertising network is faced with a database of millions of creatives that it might elect to show. The space of contexts is similarly large, consisting, for example, of all possible natural language queries in sponsored search advertising, or of all possible feature values describing a webpage in display advertising. This explosion in the number of contexts and actions make many standard algorithms computationally infeasible to implement, and furthermore lead to practically vacuous performance guarantees. In particular, the cumulative regret of any algorithm for the standard K-armed bandit problem is lower bounded by $\Omega(\sqrt{KT})$ (Auer et al., 2003).

However, this limitation can be circumvented if the payoffs are presumed to be structured, allowing some notion of similarity or generalization across actions. One of the most common assumptions is that actions are embedded in some metric space, with nearby actions generating similar rewards through a Lipschitz payoff function. This type of setting has been studied by Agrawal (1995); Kleinberg (2004); Kleinberg et al. (2008); Lu et al. (2010); Slivkins (2014), among others. Another very common assumption is that rewards are convex functions of the action selected (or are well-approximated by such a function) (Flaxman et al., 2005; Dani et al., 2007; Abernethy et al., 2008). As a special case, linear functions are expressive enough to actions that correspond to combinatorial elements such as matchings or paths in a graph Cesa-Bianchi and Lugosi (2012).

We develop a theory for a different type of structural assumption distinct from those made in the previously mentioned works. Our approach introduces a class of graphical models that

permit both the space of actions and contexts to be large, yet succinctly specify the payoffs for any pair of context and action. By analogy to probabilistic graphical models (Koller and Friedman, 2009), where missing edges represent independence assumptions on the unknown distribution, missing edges in our model limit the degree to which features corresponding to actions (or contexts) can affect payoff.

Consider a simplified example from sponsored search, where we consider keywords related to purchasing airline tickets, and would like the maximize the probability that a user accepts an advertised offer. Each search query specifies the value of two context variables: the origin city x_{origin}, and the destination city x_{dest}. Offers are described by three variables: an origin city a_{origin}, a destination city a_{dest}, and a rental car deal a_{deal}. We might assume that the expected payoff is given by an arbitrary function $F(\cdot)$ on these five variables. However, a more natural assumption for this domain might be that $F(\cdot) = f_1(x_{\text{origin}}, y_{\text{origin}}) + f_2(x_{\text{dest}}, y_{\text{dest}}) + f_3(x_{\text{dest}}, a_{\text{dest}}, a_{\text{deal}})$. In other words, f_1 and f_2 account for how well the queried cities match the advertised cities. Meanwhile, f_3 captures that how enticing the rental deal is, which might be a function of where the user wants to go (x_{dest}), and the offer's destination (a_{dest}), but would not depend on the origin cities.

We show how these domain specific assumptions can be represented by a type of graphical model, which we call the *interaction graph*. While the aim of a probabilistic graphical model is to encode independence assumptions on some distribution, we seek to encode separability assumptions in the payoff function F. Notice that in the above example, the functions $\{f_i\}$ are allowed to be arbitrary, and are not known to the learner. Only the decomposition of F into $\{f_i\}$ is known. This is analogous to assuming the structure of a probabilistic graphical model, but not the specific conditional probability tables, or distribution parameters. We also comment that since the $\{f_i\}$ are arbitrary, the payoff function F need not be convex or smooth, which as previously discussed, are the prevailing assumptions that make large scale bandits tractable.

Graphical representations have been successfully applied in the standard K-armed bandit

problem in the works of Mannor and Shamir (2011); Alon et al. (2013). However, what is called the *observability graph* in these works represents informational assumptions, rather than assumptions about structured payoffs. Specifically, nodes in an observability graph represent arms, and playing an arm a also directly reveals the payoffs of every arm connected to a. The observability graph thus interpolates between a bandit setting (characterized by the empty graph) and a full-information setting (characterized by the complete graph). We note that this differs from our setting which is distinctly a bandit setting; the payoffs of unplayed arms are never explicitly revealed. In our setting, any information gleaned about unplayed arms is done so *implicitly*, by reasoning about the interaction graph.

Our main result is an algorithm that attains $O(T^{2/3})$ regret against the best policy mapping contexts to actions. The running time of the algorithm is exponential in the tree-width of the interaction graph. Thus, if F induces an interaction graph with tree-width bounded by a constant, the algorithm runs in polynomial time. At a high level, the algorithm maintains an estimate F_{est} of the payoff function. For a fixed context, computing the best action (using the estimate F_{est}) is related to existing dynamic programming algorithms for finding the maximum a posteriori (MAP) assignment in a probabilistic graphical model. It will turn out to be crucial for the algorithm to track point-wise which inputs to F_{est} provide reliable estimates. For this, we will utilize a particular type of learner to implement F_{est}: a KWIK linear regressor. KWIK stands for "Knows What It Knows," and is a model of supervised learning wherein the learner is required to be self-aware on which inputs it can make an accurate prediction. This connection between large-scale bandits and KWIK learning can in fact be made stronger, and is the subject of the next chapter.

The results contained in this chapter first appeared in Amin et al. (2012b) with collaboration from Michael Kearns and Umar Syed.

1.2. Chapter 3: Large Scale Bandits and KWIK Learning

Continuing our study of large-scale bandit problems, we establish a deeper connection between the bandit setting and KWIK learning, which proved essential for solving the graphical bandit problem. The KWIK "Knows What it Knows" framework was introduced by Li et al. (2011) (Strehl and Littman, 2007; Walsh et al., 2009; Sayedi et al., 2010; Li and Littman, 2010), and was designed to address efficient exploration in model-free reinforcement learning (see Li and Littman (2010) in particular). Li, Littman, Walsh, and Strehl (2011) suggest machine learning problems other than reinforcement learning, including bandit problems, could benefit from the perspective introduced by KWIK learning. In this chapter, we confirm this conjecture. We show that the technique used to solve graphical bandit problems can be extended to a general reduction for bandit problems to KWIK learning. Moreover, we show that supervised learning settings weaker than KWIK do not suffice for a general reduction. In this sense, the reduction to KWIK learning is tight.

The reduction of Li and Littman (2010), holds for model-free reinforcement learning in MDPs, which generalizes many types of stochastic bandit problems. The standard K-armed bandit, for example, can be viewed as a simple MDP with K states. Thus, it's important to distinguish between their setting and ours. They reduce value-function approximation in large-state, finite-action, MDPs to KWIK learning, while our result concerns large-state, large-action, stochastic bandit problems. The most obvious distinction is that our result allows for large, or even infinite, actions. More importantly, however, is that the reinforcement learning setting assumes the learner has agency over its next state. Since its next state is in part a function of its choice of action, the reinforcement learning agent is faced with the additional challenge of planning. The contextual bandit learner, on the other hand, is excused from planning as it views its states as having been selected exogenously of its actions (as is often the case in advertising, where impressions arrive independently of what ads are delivered). However, it is faced with its own difficulty that it must be robust to non-Markovian, worst-case, state sequences.

The KWIK framework is a supervised learning model that forces that learner to track the inputs on which it can make accurate predictions. In the KWIK learning protocol, the learning problem is specified by some $F : \mathcal{Z} \to \mathcal{Y}$, where $\mathcal{Z}$ is some input space and $\mathcal{Y} \subset \mathbb{R}$, as well as accuracy parameters $\epsilon, \delta > 0$. The unknown f is assumed to belong to some parametric function class $\mathcal{F}_\Theta = \{F_\theta : \mathcal{Z} \to \mathbb{R} \mid \theta \in \Theta\}$. The learner encounters a series of observations (possibly adaptively selected by an adversary) $z^{(1)}, z^{(2)}, ...$ For each observation, the learner either predicts an output $\hat{y}^{(t)}$ or abstains by selecting $\hat{y}^{(t)} = \perp$. When not abstaining, the prediction is *required* to be accurate, with $|F_{\theta^*}(z^{(t)}) - \hat{y}^{(t)}| < \epsilon$. Otherwise, the learner has completely failed, where the probability of a failed run can be at most δ. Only when abstaining from making a prediction does the learner receive feedback, in the form of noisy estimates of $F_{\theta^*}(z^{(t)})$. We say $\mathcal{F}_\Theta$ is KWIK learnable if there exists an algorithm for the above protocol that (1) runs in polynomial time, (2) abstains from prediction on at most poly$(\epsilon^{-1}, \delta^{-1})$ rounds.

A stochastic bandit problem is also instantiated by some function $F : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ mapping a state x, and choice of action a to an expected reward $\mathbb{R}$. As discussed in the previous chapter, without structural assumptions on F, efficient no-regret is impossible. Thus, we take F to belong to some parametric class $\mathcal{F}_\Theta = \{F_\theta : \mathcal{X} \times \mathcal{A} \to \mathbb{R} \mid \theta \in \Theta\}$. In this chapter we established that the large-scale parametric bandit problem can be efficiently reduced to KWIK learning $\mathcal{F}_\Theta$, (where $\mathcal{Z} = \mathcal{X} \times \mathcal{A}$).

The KWIK protocol can be understood as combining elements of PAC-learning Valiant (1984) and the Mistake Bound (MB) model Littlestone (1988) (see Li et al. (2011)). Similar to an MB algorithm, a KWIK algorithm makes a bounded number of mistakes, but must announce on which rounds those mistakes could occur. Although a PAC algorithm need only consider iid inputs, it can be thought as abstaining from labels during training, so that it can make accurate predictions thereafter. Any KWIK algorithm can be converted to an MB algorithm for the same function class, simply by making arbitrary predictions instead of abstaining. Similarly, MB algorithms can be used to provide PAC algorithms (Blum,

1994).

KWIK learnability is therefore a rather strong condition. One natural question is whether other, weaker, notions of supervised learning suffice for a general reduction. Our second main result in this chapter, addresses this question. We give a matching intractability result showing that the demand for KWIK learnability is necessary, in that it cannot be replaced with standard online no-regret supervised learning, or weaker models such as PAC learning, while still implying a solution to the MAB problem. We show a general reduction based on a weaker model would admit the inversion of cryptographic trapdoor functions. Thus our reduction is thus tight with respect to the necessity of the KWIK learning assumption.

The results contained in this chapter first appeared in Abernethy et al. (2013) with collaboration from Jacob Abernethy, Moez Draief, and Michael Kearns.

1.3. Chapter 4: Large Scale Bandits : Lower Bounds on Regret

We next introduce a notion of complexity for large-scale parametric bandits, our object of study in the previous two chapters. As previously mentioned, there is a great deal of prior work on large-scale bandits that admit some structural assumptions. For example, in the case where the learner is presented with an L-Lipschitz function, Kleinberg et al. (2008) give an algorithm that attains sublinear regret. The rate at which their algorithm attains no-regret, however, depends on a measure of complexity they call the *zooming dimension*. More benign functions admit a smaller zooming dimension, and better rates. In this Chapter we explore whether a notion of complexity can be derived for *arbitrary* classes $\mathcal{F}$. Given an instance of a parametric MAB problem, $\mathcal{F}$, can we characterize the optimal regret rates for that problem?

We consider the stochastic multi-armed bandit for a general parametric class $\mathcal{F}$. For these results, we limit our analysis to the non-contextual version of the problem, where payoffs depend only on the choice of action; for some unknown $\mathcal{F} \subseteq \{F_\theta : \mathcal{A} \to \mathbb{R}\}$, and the learner observes noisy estimates of $F(a^{(t)})$ for each choice of action $a^{(1)}, ..., a^{(t)}$. Our main

contribution is defining a measure of complexity of $\mathcal{F}$ that we call the *haystack dimension* of $\mathcal{F}$ (denoted $HD(\mathcal{F})$). Intuitively, the haystack dimension captures the extent to which maximizing a function via queries requires a search for a small number of items (needles) amongst a much larger number of otherwise undifferentiated possibilities (a haystack). We give upper and lower bounds on regret involving this quantity, that are nearly tight (within log factors of each other). Compared to prior work, our results provide a complete analysis for significantly more general function classes.

Our characterization holds for any finite-cardinality $\mathcal{F}$ (though the number of *actions* may still be infinite), but we also describe a generalization to the case of infinite $\mathcal{F}$ via covering techniques (where the tightness of the bounds depends on checking a certain technical condition). Our lower bound is also information theoretic, ignoring computational complexity. In this sense, the haystack dimension can be seen as playing a role in the study of functional MAB problems analogous to that played by quantities such as VC dimension (Vapnik and Chervonenkis, 1971) and teaching dimension (Goldman and Kearns, 1995) in other learning models, which also characterize sample or informational complexity, but not computational complexity.

An interesting aspect of our methods is the connection drawn between MAB problems and the problem of exact learning of functions from queries. We observe that functional MAB problems implicitly embed the problem of finding a maximum of an unknown function in $\mathcal{F}$ from only input-output queries (generalizations of membership queries), which may or may not be much easier than exact learning. Hegedüs (1995) characterized the query complexity of exact learning in terms of the *extended teaching dimension* of the function class. For some restricted function classes the haystack dimension and extended teaching dimension coincide, and in these cases our analysis approximately recovers the bounds due to Hegedüs (1995), but with a significant advantage: our lower bound holds for all *randomized* algorithms, while the earlier bound only applied to deterministic algorithms. A variant of exact learning of functions has been considered under the name of generalized

binary search. Nowak (2009) introduces a notion of dimensionality that is in some ways similar to ours, but once again only addresses the difficultly of exact learning, and does not address the maximization problem and its relationship to MAB. To our knowledge, the only other work to introduce a measure of complexity for MAB problems parameterized by some *general* function class is Russo and Van Roy (2013). However, their measure of complexity is used only to give an upper bound for a Thompson sampling algorithm. The eluder dimension does not appear to lead to a lower bound, and thus does not necessarily characterize the complexity of *any* algorithm for the MAB problem on $\mathcal{F}$.

The results contained in this chapter first appeared in Amin et al. (2011) with collaboration from Michael Kearns and Umar Syed.

1.4. Chapter 5: Posted Price Auctions Against Strategic Buyers

Until now we have discussed settings where the reward of the decision-maker is a stochastic function of it's choice of action and the context in which that action is taken. In a liquid market, this is usually a very reasonable assumption. For example, if there are many advertisers interested in a single impression, no single advertiser's behavior has a large effect on the market, and we can take a macroscopic viewpoint where strategic behavior on the part of advertisers in essentially ignored.

In this chapter we consider an important setting in which we can no longer take this viewpoint. Namely, if there are relatively few advertisers (the buyers) competing for a particular type of impression, the advertising network (the seller) must take the buyers' strategic considerations into account. In other words, the reward sequences received by the seller are now tightly coupled with the underlying mechanism that is pricing impressions, as well as the seller's choice of algorithm.

The prevailing method for allocating advertising to impressions is the second-price auction. Once the seller has determined which advertisers will participate in an auction, there are several parameters which determine the details of the auction, and which might have large

effects on revenue. One of the most basic choices in a second price auction is the selection of *reserve price* (or floor price), a minimum asking price for the impression. If no advertiser's bid clears the reserve price, the impression remains unsold. Otherwise, the winner of the auction pays the maximum of the second-highest bid and the reserve price.

Intuitively, the selection of reserve price becomes increasingly important as fewer advertisers compete for an impression. In the extreme case where an impression attracts only a single advertiser, for example, the ad networks revenue hinges entirely on it's choice of reserve price. This extreme case, in fact, happens rather frequently. One of the great advantages of online advertising, compared to advertising in traditional media, is the presence of rich real-time information about the impression. This information is a boon to advertisers, allowing them to bid on only those impressions that they value. However, such precise targeting has the unfortunate side-effect that some impressions attract very few advertisers. Second price auctions with a single bidder and reserve are equivalent to posted price auctions (the reserve price is announced and the advertiser either accepts or rejects the offer by over- or under-bidding the reserve). We therefore analyze the problem of maximizing an auctioneer's revenue in a repeated posted price auction.

Posted price auctions have a long history of study, including more recent literature motivated by internet auctions (Bar-Yossef et al., 2002; Blum et al., 2003; Kleinberg and Leighton, 2003). Unlike in these previous works, which assume a monopolistic seller is faced with a sequence of *different buyers*, we suppose that the same seller is repeatedly interacting with the *same buyer* across multiple rounds. Since the buyer's action on any particular round might influence the prices offered on later rounds, the buyer might be incentivized to conceal its true value in the hope for better prices in the future. Furthermore, and perhaps unsurprisingly, there is empirical evidence suggesting that buyers do indeed behave strategically when bidding in advertising auctions (Edelman and Ostrovsky, 2007).

Repeated posted price actions against the same strategic buyer have been considered in the economics literature under the heading of *behavior-based price discrimination* (BBPD)

by Hart and Tirole (1988); Schmidt (1993); Acquisti and Varian (2005); Fudenberg and Villas-Boas (2006), and more recently by Devanur et al. (2014). Our setting differs from BBPD in two key ways.

First, all these works imagine that the buyer's type is drawn from some fixed publicly available distribution. Assuming that the seller knows this distribution, but not the realized buyer type, the appropriate solution concept in these works is identifying strategy profiles that constitute perfect Bayesian equilibria. This assumption, that there is some known distribution generating buyer types, has been common in the economics literature ever since Myerson (1981). However, this introduces a philosophical question of how the seller arrived at this distribution to begin with. Since the buyer is incentivized to hide its true value to avoid a discriminating seller, it's unclear why the seller would have ever had the opportunity to learn a good estimate — indeed, the correct estimate — of this distribution. In this sense, the BBPD approach takes for granted that some non-trivial learning has occurred at the start of the game. We treat the learning problem directly by supposing that the buyer's value for the item on each round is drawn from some fixed distribution that is *unknown* to the seller at the start of the game. Our goal is to attain the Myerson optimal revenue (in the limit of the number of auctions) against a buyer who plays a best response.

Secondly, we assume that the seller publicly commits to a pricing strategy to which the buyer plays a best-response. This is in contrast to the solution concepts in the BBPD, which depend highly on the seller's lack of commitment to future prices (Fudenberg and Villas-Boas, 2006). The motivation for our model comes from internet advertising where a common complaint levied by buyers against dynamic floor pricing is the lack of transparency. We seek a strategy which can be publicly announced, and to which the seller will truthfully adhere, but that generates a large amount of revenue for the seller.

To understand the difference between these two settings, it's illustrative to consider two simple strategies. Consider a seller who wishes to set the Myerson optimal price on every

round, and a buyer who wishes to behave truthfully (accepting prices iff they are above his value). In our setting, this seller strategy is simply infeasible: the distribution is unknown to the seller. In the BBPD literature, these two strategies do not constitute a perfect Bayesian equilibrium. Should the seller reject the first offer, the buyer will no longer wish to play his strategy in the resulting sub-game, but would prefer to set a lower price. This latter issue does not arise in our setting, as the seller moves first and commits to a strategy for the entire game.

We note that in many important real-world markets and particularly in online advertising that sellers are far more willing to wait for revenue than buyers are willing to wait for goods. For example, advertisers are often interested in showing ads to users who have recently viewed their products online (this practice is called retargeting), and the value of these user impressions decays rapidly over time. Or consider an advertising campaign that is tied to a product launch. A user impression that is purchased long after the launch (such as the release of a movie) is almost worthless. Our results are qualitatively similar to those in BBPD (Fudenberg and Villas-Boas, 2006), and depend on this relationship between buyer and seller discount factors.

If the seller is more patient than the buyer (i.e. is less severely discounted), we give an algorithm that attains the Myerson optimal revenue in the limit. However, assuming that the discount factors are the same, we show that achieving this goal is impossible. In particular, the seller is best off playing an oblivious strategy that never attempts to learn from past observations.

The results contained in this chapter first appeared in Amin et al. (2013) with collaboration from Afshin Rostamizadeh and Umar Syed.

1.5. Chapter 6: Advertiser Budget Optimization in Keyword Auctions

Sponsored search advertising is one of the most common types of online advertising. A search engine, whose users input textual strings indicating the content that they desire

to see, responds with so-called organic results (which are not influenced by advertising interests), as well as paid sponsored advertising relevant to the user's query. Advertisers interested in a particular keyword are automatically entered into a second-price auction whenever a user searches for that keyword.

In this chapter, we take the perspective of an advertiser, seeking to maximize the value received from running a keyword auction campaign. Typically, the advertiser will specify a budget that it is willing to spend over a period of time: a single day, week, month, etc. The advertiser therefore wishes to bid in a manner that maximizes the value of clicks attained for the specified budget during such a period. When the period ends, the advertisers budget is reset for the next period.

This budget optimization problem is closely related to the online knapsack optimization. Consider the case of single-slot auctions, where only the highest bidder on each round displays its advertisement. If one takes the perspective of a particular advertiser, then the largest *competing bid* represents the rest of the market, and amount that the advertiser would have to pay in order to win the impression. As observed by Borgs et al. (2007), and later by Zhou et al. (2008), this is easily modeled as an instance of online knapsack: the advertiser is presented with a sequence (v_t, p_t) representing the value and the *market price* (the largest bid among competing advertisers) of each impression. During a period, the bidder must decide which impressions to acquire for its budget, and these decisions must be made in an online fashion (on round t, decisions made before round t cannot be altered).

The online knapsack problem is known to be a hard problem. In particular, for any constant $C > 1$, there is no algorithm that attains a competitive ratio of C against the offline omniscient algorithm (which knows the sequence in advance and computes the offline optimal) (Marchetti-Spaccamela and Vercellis, 1995). Nevertheless, Zhou et al. (2008) give an algorithm which attains a non-trivial competitive ratio as long as the value-to-price ratio of items are upper and lower bounded. They show a competitive ratio of $\ln(U/L) + 1$, where $L <= v_t/p_t <= U$, with a matching lower bound for any algorithm.

The assumption of arbitrary sequences in these works, however, is quite strong. While advertisers may bid strategically, the nature of the interaction between advertisers is not zero-sum, and therefore an individual advertiser should not expect to be presented with a "worst-case" sequence of competing bids. In fact, in a crowded market, the prices p_t might even begin to look stochastic from a macroeconomic perspective.

We will make precisely this assumption, that the competing bid p_t is drawn from a fixed but unknown probability distribution. We view this as analogous to classical models in finance, where despite strategic behavior of traders at the individual level, models of macroscopic price evolution that are stochastic (such as Brownian motion models) have been quite effective in developing both models and algorithms. Indeed, our resulting algorithm is similar in spirit to an algorithm of Ganchev et al. (2010). for executing trades in dark pools, a type of limited-information security exchange.

The stochastic relaxation of the online knapsack problem has been previously considered, by (Marchetti-Spaccamela and Vercellis, 1995), and later by Lueker (1995). Lueker (1995) presents an efficient algorithm, the value of its solution *differing*[1] from the offline optimal by $\Theta(\ln T)$ on a sequence of T items. Lueker's algorithm, however, assumes that the distribution generating the knapsack items is known exactly (there is no learning component to the problem).

This algorithm is applied by Zhou and Naroditskiy (2008) to the problem of keyword bidding by estimating statistics sufficient to run Lueker's algorithm from observed data. This approach, however, highlights a shortcoming in treating the problem of bidding in keyword auctions as a straightforward online knapsack problem. In a keyword auction, the competing bid p_t is not revealed to the advertiser at the beginning of the round (it is, after all, a function of the private bids of all the advertiser's competitors). Thus, the sequence (v_t, p_t)

[1]In contrast to previous work, this is an asymptotic statement about regret, rather than competitive ratio. Lueker's model assumes that the knapsack's capacity grows with the number of items. Under such an assumption, the value of the optimal solution is $\Theta(T)$, and $\Theta(\log T)$ regret implies competitive ratios tending to 1.

is not so simply observed, at least not directly. An advertiser instead receives censored observations of p_t. In particular, by bidding b_t, an advertiser who fails to win an impression only gains some information, that $b_t < p_t$.

The algorithms described in this work take advantage of the literature on censored learning, as well as the assumption of stochastic competing prices, to give state-of-the-art empirical performance on real keyword auction data acquired from Microsoft adCenter. Our performance on real data demonstrates the value of treating market prices stochastically. These algorithms, GreedyProductLimit, and LuekerLearn (which uses Lueker's stochastic knapsack algorithm as a subroutine), were later shown by Tran-Thanh et al. (2014) to be Hannan consistent, with a regret rate of $O(\sqrt{T})$.

The results contained in this chapter first appeared in Amin et al. (2012a) with collaboration from Michael Kearns, Peter Key and Anton Schwaighofer.

CHAPTER 2 : Graphical Bandits and the Ad-Selection Problem

In this chapter, we introduce a rich class of graphical models for multi-armed bandit (MAB) problems that permit both the state or context space and the action space to be very large, yet succinctly specify the payoffs for any context-action pair. In our models there may be many context state variables, whose values are chosen exogenously by Nature, as well as many action variables, whose values must be chosen by a MAB algorithm. Only when all context and action variables are assigned values is an overall payoff observed. Thus each setting of the context variables yields a distinct but related bandit problem in which the number of actions scales exponentially.

As discussed in the introduction, settings where the number of contexts and actions are both large are becoming common in applied settings such as sponsored search, where the number of possible user queries is effectively unbounded, and on many frequent queries the number of possible ads to show may also be large. Similarly, in many problems in quantitative trading, the number of ways one might break a trade up over multiple exchanges (the action space) is large, but might also depend on many conditioning variables such as market volatility (the context space). Recent lines of research have considered large parameterized context spaces and large parameterized action spaces separately (see Related Work below); here we are interested in models and algorithms for their simultaneous presence.

We consider a setting in which an MAB algorithm knows the coarse underlying graphical or dependency structure of the model, but not the actual parameters. Our main result is a no-regret algorithm for solving graphical MAB problems that is computationally efficient for a wide class of natural graphical structures. By no-regret we mean that the total regret of our algorithm is bounded by the number of parameters of the model, and thus is generally greatly sublinear in the number of contexts and actions. Interestingly, the running time of our algorithm depends only on properties of the substructure of the graphical model induced by the action variables — the dependencies between context variables, or between context

and action variables, may be arbitrarily complex. If the treewidth of the action subgraph is constant, our algorithm runs in polynomial time.

The algorithm itself combines distributed dynamic programming — where the main challenge we face compared to standard such computations is the lack of any payoff until the entire model is instantiated (no local observations) — with the fact that our models admit linearization via a certain natural vector space of coefficients, which permits the application of recent "Knows What It Knows" (KWIK) algorithms for noisy linear regression. At a high level, for each context chosen by Nature, either our algorithm succeeds in choosing a reward-maximizing action (an exploitation step), or it effectively discovers another basis vector in a low-dimensional linear system determined by the graphical model (an exploration step).

The regret of the algorithm depends on the rank of this underlying vector space, which is always bounded by the number of parameters but may be smaller. It is a feature of our algorithm that no distributional assumptions are needed on the sequence of contexts chosen, which may be arbitrary. However, in these case that a distribution is present, the effective rank and thus our regret may be smaller still.

2.1. Related Work

Many authors have studied bandit problems where the number of states or *contexts* (each of which indexes a separate but perhaps related bandit problem) is allowed to be large Auer et al. (2003); Langford and Zhang (2007); Beygelzimer et al. (2011), while the number of actions, or the size of the function class to which the payoff function belongs, are assumed to be small (i.e. the sample and computational complexity of these algorithms are allowed to depend linearly on either of these quantities). In contrast, our results will consider both payoff function classes that are infinite, as well as context and action spaces that are assumed to be very large and thus call for sublinear complexity.

As we will demonstrate in Section 2.5, the setting we consider can be thought of as a linearly parameterized bandit problem. Such models associate each action $\mathbf{x}$ with a feature vector

$\phi(\mathbf{x})$, and the payoff for taking that action is given by $\phi(\mathbf{x}) \cdot \mathbf{w}$, where $\mathbf{w}$ is an unknown weight vector. The computational complexity of most existing algorithms is nevertheless linear in the number of actions Abe et al. (2003); Auer (2002); Li et al. (2010). Furthermore, rather than being specified explicitly, the linear space in which our parameterizations lie are given by the underlying graphical or locality structure of the model.

As discussed in the introduction (c.f. Section 1.1), there are structural assumptions which admit efficient algorithms when both action and context spaces are infinite. These include convexity assumptions (Flaxman et al., 2005; Dani et al., 2007; Abernethy et al., 2008), and metric similarity assumptions (Agrawal, 1995; Kleinberg, 2004; Kleinberg et al., 2008; Lu et al., 2010; Slivkins, 2014). We view this work as introducing a key set of assumptions missing from the literature: that dependencies between actions admit a certain graphical representation.

While the results of these settings and ours are not directly comparable, as each permits different rich classes of functions, we review the regret bounds that are ultimately attained. The regret bounds given by both Slivkins (2014) and Lu et al. (2010) are $O(T(1+d)/(2+d))$ where d is some dimensionality constant dependent on the action and context spaces. In contrast, we provide regret bounds that are $\tilde{O}(T^{2/3})$, and leave as an interesting open question whether such a rate can be improved. Convexity assumptions yield better $O(\sqrt{T})$ bounds (Abernethy et al., 2008). However, we note that the algorithm which attains the optimal rate is slightly involved, requiring a self-concordant barrier function for the comparator class of convex functions. In contrast, our algorithm is quite natural relying on dynamic programming and noisy linear regression.

2.2. Preliminaries

We begin by assuming that both actions and contexts are represented by vectors of discrete variables, and that there is an unknown function which maps assignments to these variables to a payoff. The graphical assumption of our model will come when we assume how these

variables interact.

Let $F : \prod_{i \in V} X_i \to [0, 1]$ be the unknown *expected payoff function*, where each X_i is the set of possible *values* for *variable* $i \in V$, with $|X_i| \geq 2$. The set of variables is partitioned into a subset $A \subseteq V$ of *action variables* and a subset $C \subseteq V$ (disjoint from A) of *context variables*, with $V = A \cup C$. Let $m = \max_i |X_i|$ and $n = |V|$.

For any subset of variables $S \subseteq V$, let $\mathbf{x}_S \in \prod_{i \in S} X_i$ denote a *joint assignment* of values to variables S. For shorthand, we write $\mathbf{X}_S = \prod_{i \in S} X_i$ to denote the set of all possible joint assignments to variables S. When $S = V$, we typically drop the subscript and write $\mathbf{x}$ and $\mathbf{X}$ instead of $\mathbf{x}_V$ and $\mathbf{X}_V$, and call $\mathbf{x}$ a *complete joint assignment*. We also abbreviate $\mathbf{x}_{\{i\}}$ as $\mathbf{x}_i$, when referring the assignment of a single variable. If $S = A$ we call $\mathbf{x}_S$ a *joint action*, and if $S = C$ we call $\mathbf{x}_S$ a *joint context*.

2.2.1. Learning Protocol

Learning in our model proceeds as a series of trials. On each trial, Nature determines the current state or context, and our algorithm must choose values for the action variables in order to complete the joint assignment. Only then is a reward obtained, which depends on both the context and action.

In each round $t = 1, \ldots, T$:

1. Nature chooses an arbitrary joint context assignment $\mathbf{x}_C^t$, which is observed by the learning algorithm.

2. The learning algorithm chooses a joint action assignment $\mathbf{x}_A^t$.

3. The learning algorithm receives an independent random payoff $f^t \in [0, 1]$ with expectation $F(\mathbf{x}_A^t, \mathbf{x}_C^t)$.

The *regret* after T rounds is

$$R(T) \triangleq E\left[\sum_{t=1}^{T} \max_{\mathbf{x}_A} F(\mathbf{x}_A, \mathbf{x}_C^t) - F(\mathbf{x}_A^t, \mathbf{x}_C^t)\right]$$

where the expectation is with respect to randomness in the payoffs and the algorithm.

Note that in the learning protocol above, Nature chooses the context assignments as an arbitrary sequence; our main results hold for this rather strong setting. However, in Section 2.6.1, we also consider the special case in which each joint context $\mathbf{x}_C^t$ is drawn from a fixed distribution $\mathcal{D}$ (which is also the assumption of much of the previous research on contextual bandits), where better bounds may be obtained.

2.2.2. Assumption on Payoff Function

The main assumption that we leverage for our results is that, while F is unknown, we know that certain sets of variables may interact with each other to affect the payoff, while other sets may not. This is made precise in Assumption 2.1 below.

Assumption 2.1 (Payoff Decomposition)**.** *We are given a collection of variable subsets $\mathcal{P} \subseteq 2^V$ such that the unknown payoff function F has the form*

$$F(\mathbf{x}) = \sum_{P \in \mathcal{P}} f_P(\mathbf{x}_P)$$

where each unknown function $f_P : \mathbf{X}_P \to \mathbb{R}$ is called a potential function*.*

We emphasize that the potential functions are unknown and arbitrary. What is known is that F admits such a decomposition; more precisely, $\mathcal{P}$ is known.

Note that Assumption 2.1 is without loss of generality, since we can always take $\mathcal{P} = \{V\}$ and $f_V = F$. However, we are primarily interested in settings where F decomposes into much simpler potential functions. If $|P| \leq k$ for all $P \in \mathcal{P}$ then we say the potential functions are *k-ary*.

Also note that Assumption 2.1 is very similar to the kinds of assumptions often made for tractable approximate inference (such as in a Markov random field), where a complex multivariate distribution is assumed to factorize into a product of several potential functions, each of which depend on only a subset of the variables. We next elaborate on the graph-theoretic aspects of our model.

2.2.3. Interaction Graphs

In what follows, we will use structural properties of the expected payoff function F to bound the regret and running time of our graphical bandit algorithm. In particular, our results will depend on properties of the *interaction graph* G of the expected payoff function F. Let $G = (V, E)$ be a graph on variables V with edges E defined as follows: For any pair of variables $i, i' \in V$ we have edge $\{i, i'\} \in E$ if and only if there exists $P \in \mathcal{P}$ such that $i, i' \in P$, where $\mathcal{P}$ was defined in Assumption 2.1. In other words, we join i and i' by an edge if and only if there is some potential function f_P that depends jointly on the variables i and i'. The *action subgraph* $G_A = (A, E_A)$ is the subgraph of G containing only the action variables $A \subseteq V$ and the edges between them $E_A \subseteq E$.

Note that the relationship between the interaction graph G and an expected payoff function F is essentially analogous to the relationship between a graphical model and a distribution whose independence structure it represents. The absence of an edge between two variables in a graphical model indicates that the variables are *probabilistically independent* in the distribution, while the absence of an edge between two variables in an interaction graph indicates that the two variables are *separable* in the expected payoff function. A sparse graphical model leads to computationally tractable inference algorithms, and we will shortly see that a sparse interaction graphs (more precisely, sparse action subgraphs) lead to computationally tractable no-regret algorithms for a graphical bandit problem.

2.3. Algorithm Overview

We present the details of our graphical bandit algorithm over the next three sections, and give a high-level overview in this section. Our approach will be to divide the graphical bandit problem into two subproblems — one essentially dealing with exploitation, and the other with exploration — and then compose the solutions to those subproblems into a no-regret algorithm.

In Section 2.4, we describe the `BestAct` algorithm, which, for any given joint context $\mathbf{x}_C$, computes an ϵ-optimal joint action $\mathbf{x}_A^\epsilon$. The `BestAct` algorithm assumes access to an ϵ-good approximation $F^\epsilon(\cdot, \mathbf{x}_C)$ of the expected payoff $F(\cdot, \mathbf{x}_C)$, and when computing $\mathbf{x}_A^\epsilon$ for given joint context $\mathbf{x}_C$ the `BestAct` algorithm makes many calls to the oracle $F^\epsilon(\cdot, \mathbf{x}_C)$. Note that `BestAct` cannot simply issue these queries to Nature instead, since it has no way of fixing the joint context $\mathbf{x}_C$ over several rounds. The running time of `BestAct` is polynomially bounded if the treewidth of the action subgraph G_A is constant. `BestAct` is related to existing dynamic programming algorithms for finding the maximum *a posteriori* (MAP) assignment in a graphical model, but unlike that setting must overcome the additional challenge of receiving only *global* payoffs and no direct local observations or information.

In Section 2.5, we describe the `PayEst` algorithm, which approximately implements the oracle required by `BestAct`. With high probability, whenever `PayEst` receives a complete joint assignment $(\mathbf{x}_A, \mathbf{x}_C)$, either it outputs the value of $F^\epsilon(\mathbf{x}_A, \mathbf{x}_C)$, or it outputs a special symbol $\perp$. `PayEst` is an instance of a "Knows What It Knows" (KWIK) algorithm Li et al. (2011), and the number of times `PayEst` outputs $\perp$ is polynomially-bounded if the potential functions are all k-ary for a constant k. The bound is a consequence of the fact that F can be written as a linear function, and each $\perp$ output increments the dimension of a certain linear subspace. Again, note that `PayEst` cannot be as simple as repeatedly playing the complete joint assignment $(\mathbf{x}_A, \mathbf{x}_C)$ and averaging the observed payoffs, since our algorithm can only specify the joint action $\mathbf{x}_A$ in each round, and has no way of fixing the joint context $\mathbf{x}_C$.

In Section 2.6, we put `BestAct` and `PayEst` together to form `GraphicalBandit`, an algorithm for graphical bandit problems, which in each round t runs `BestAct` for the current context $\mathbf{x}_C^t$, and uses `PayEst` to approximately implement the oracle required by `BestAct`. The main difficulty in integrating the two algorithms is that sometimes `PayEst` outputs $\perp$ instead of the value of the oracle $F^\epsilon(\cdot, \mathbf{x}_C^t)$. However, whenever this happens, `BestAct` will provide feedback that causes `PayEst` to make measurable exploration progress, improving its ability to provide accurate payoff estimates in subsequent rounds.

2.4. Best Action Subroutine

We will now describe `BestAct`, an algorithm that efficiently computes, for a given joint context $\mathbf{x}_C$, an ϵ-optimal joint action $\mathbf{x}_A^\epsilon$ satisfying $F(\mathbf{x}_A^\epsilon, \mathbf{x}_C) \geq \max_{\mathbf{x}_A} F(\mathbf{x}_A, \mathbf{x}_C) - \epsilon$. The `BestAct` algorithm uses dynamic programming applied to the action subgraph G_A, and is similar to standard dynamic programming algorithms, such as the Viterbi algorithm, for computing the MAP assignment in a graphical model. Our setting is more challenging, however. For one, we do not have access to the individual potential functions f_P, but only to the expected payoff function F (assuming no noise). Furthermore, we do not control the context argument to F.

To illustrate these difficulties, suppose that the action subgraph G_A is a tree, and that we wish to run a standard dynamic programming algorithm which, fixing an assignment for a variable's parent, attempts to find the best assignment for the subtree rooted at that variable, under a joint context $\mathbf{x}_C$. That algorithm would require several queries to $F(\cdot, \mathbf{x}_C)$. However, once the first observation is received, the next context is not guaranteed to be the same, and the algorithm suffers regret for that round.

Furthermore, as we have previously remarked, any observation $F(\mathbf{x}_A, \mathbf{x}_C)$, does not give us the values of the individual potentials $f_P(\mathbf{x}_P)$. Thus, to compare two joint assignments to a subtree in the algorithm sketched, we must be careful to fix the assignment to all variables outside the subtree in question. This becomes even more delicate when G_A is not a tree.

In order to overcome these problems, we will assume oracle access to a function $F^\epsilon(\cdot, \mathbf{x}_C)$ for the given joint context $\mathbf{x}_C$ that satisfies $|F^\epsilon(\mathbf{x}_A, \mathbf{x}_C) - F(\mathbf{x}_A, \mathbf{x}_C)| \leq \epsilon$ for all joint actions $\mathbf{x}_A$. In Section 2.5 we describe the `PayEst` algorithm, which approximately implements this oracle.

For ease of exposition, we only give a detailed algorithm and analysis for the case when G_A is acyclic, although we have a related algorithm that can be applied when G_A is an arbitrary graph. We state our results for arbitrary graphs at the end of this section, and defer their details to Section 2.8.

Suppose that the action subgraph G_A is a tree. (Note that this implies that each potential function f_P depends jointly on at most two variables). Let r be an arbitrarily chosen root for G_A. For any $a \in A$, let $T(a)$ denote the vertices of the subtree rooted at a. Let $ch(a)$ denote the set of vertices which are children of a, and for $a \neq r$, let $pa(a)$ denote the parent of a. Also for $a \neq r$, let $R(a) = A \setminus (T(a) \cup \{pa(a)\})$ be remaining vertices that are neither $pa(a)$ nor belong to $T(a)$.

If $S_1, \ldots, S_k \in 2^V$ are mutually disjoint subsets of variables such that $S = S_1 \cup \cdots \cup S_k$, we write a joint assignment $\mathbf{x}_S$ as $\mathbf{x}_S = (\mathbf{x}_{S_1}, \ldots, \mathbf{x}_{S_k})$.

For $a \neq r$, we would like to compute the best joint assignment to the variables of $T(a)$, having fixed an assignment for $pa(a)$. We will denote this best joint assignment by $[a, \mathbf{x}_{pa(a)}]^*$. For any leaf a and assignment $\mathbf{x}_{pa(a)}$,

$$[a, \mathbf{x}_{pa(a)}]^* = \arg\max_{\mathbf{x}_a} F(\mathbf{x}_a, \mathbf{x}_{pa(a)}, \mathbf{x}'_{R(a)}, \mathbf{x}_C)$$

where $\mathbf{x}'_{R(a)}$ is a fixed, but arbitrary joint assignment to the vertices in $R(a)$. For a fixed choice of $\mathbf{x}_{pa(a)}$, $[a, \mathbf{x}_{pa(a)}]^*$ is the same regardless of how $\mathbf{x}'_{R(a)}$ is chosen since, by assumption, we can write $F(\mathbf{x}_a, \mathbf{x}_{pa(a)}, \mathbf{x}'_{R(a)}, \mathbf{x}_C) = f_{\{pa(a),a\}}(\mathbf{x}_{pa(a)}, \mathbf{x}_a, \mathbf{x}_C) + \sum_{e \in E_A \setminus \{(p(a),a)\}} f_e(\mathbf{x}_p, \mathbf{x}_C)$, where second term is a constant with respect to $\mathbf{x}_a$. If we had access to $F(\cdot, \mathbf{x}_C)$, $[a, \mathbf{x}_{pa(a)}]^*$ can be efficiently computed.

In general, for $a \neq r$ not necessarily a leaf node, and $ch(a) = \{a_1, ..., a_d\}$, we can write:

$$\mathbf{x}^*_{a,\mathbf{x}_{pa(a)}} = \arg\max_{\mathbf{x}_a} F(\mathbf{x}_a, [a_1, \mathbf{x}_a]^*, ..., [a_d, \mathbf{x}_a]^*, \mathbf{x}_{pa(a)}, \mathbf{x}'_{R(a)}, \mathbf{x}_C)$$

and

$$[a, \mathbf{x}_{pa(a)}]^* = (\mathbf{x}^*_{a,\mathbf{x}_{pa(a)}}, [a_1, \mathbf{x}^*_{a,\mathbf{x}_{pa(a)}}]^*, ..., [a_d, \mathbf{x}^*_{a,\mathbf{x}_{pa(a)}}]^*)$$

This motivates an algorithm, wherein the values $[a, \mathbf{x}_{pa(a)}]^*$ are computed for each a and all choices $\mathbf{x}_{pa(a)}$. This can be done efficiently by taking a in postfix order (children before parents).

However, we only have access to the approximate oracle $F^\epsilon(\cdot, \mathbf{x}_C)$. Therefore, we will instead consider:

$$\mathbf{x}^{*,\epsilon}_{a,\mathbf{x}_{pa(a)}} = \arg\max_{\mathbf{x}_a} F^\epsilon(\mathbf{x}_a, [a_1, \mathbf{x}_a]^\epsilon, ..., [a_d, \mathbf{x}_a]^\epsilon, \mathbf{x}_{pa(a)}, \mathbf{x}'_{R(a)}, \mathbf{x}_C)$$

and

$$[a, \mathbf{x}_{pa(a)}]^\epsilon = (\mathbf{x}^{*,\epsilon}_{a,\mathbf{x}_{pa(a)}}, [a_1, \mathbf{x}^{*,\epsilon}_{a,\mathbf{x}_{pa(a)}}]^\epsilon, ..., [a_d, \mathbf{x}^{*,\epsilon}_{a,\mathbf{x}_{pa(a)}}]^\epsilon)$$

where each $\mathbf{x}'_{R(a)}$ is selected arbitrarily. We now argue that the $[a, \mathbf{x}_{pa(a)}]^\epsilon$ values, computed using the function $F^\epsilon(\cdot, \mathbf{x}_C)$, are good values with respect to the true payoff function $F(\cdot, \mathbf{x}_C)$.

Theorem 2.1. *For any action variable $a \in A$, and assignments $\mathbf{x}_{pa(a)}$, $\mathbf{x}'_{R(a)}$,*

$$\begin{aligned} &F(\mathbf{x}_{pa(a)}, [a, \mathbf{x}_{pa(a)}]^*, \mathbf{x}'_{R(a)}, \mathbf{x}_C) \\ &\leq F(\mathbf{x}_{pa(a)}, [a, \mathbf{x}_{pa(a)}]^\epsilon, \mathbf{x}'_{R(a)}, \mathbf{x}_C) + 2|T(a)|\epsilon \end{aligned}$$

Proof. Fix a choice of $\mathbf{x}'_{R(a)}$ and $\mathbf{x}_{pa(a)}$. Let a be a leaf. We have that:

$$\begin{aligned} &F([a, \mathbf{x}_{pa(a)}]^*, \mathbf{x}_{pa(a)}, \mathbf{x}'_{R(a)}, \mathbf{x}_C) - \epsilon \\ &\leq F^\epsilon([a, \mathbf{x}_{pa(a)}]^*, \mathbf{x}_{pa(a)}, \mathbf{x}'_{R(a)}, \mathbf{x}_C) \\ &\leq F^\epsilon([a, \mathbf{x}_{pa(a)}]^\epsilon, \mathbf{x}_{pa(a)}, \mathbf{x}'_{R(a)}, \mathbf{x}_C) \\ &\leq F([a, \mathbf{x}_{pa(a)}]^\epsilon, \mathbf{x}_{pa(a)}, \mathbf{x}'_{R(a)}, \mathbf{x}_C) + \epsilon \end{aligned}$$

Rearranging establishes the inequality. Now suppose that the claim is true for each $a' \in ch(a) = \{a_1, ..., a_d\}$ for some variable a and assignment $\mathbf{x}_{p(a)}$.

$$F([a, \mathbf{x}_{pa(a)}]^*, \mathbf{x}_{pa(a)}, \mathbf{x}'_{R(a)}, \mathbf{x}_C) = F(x^*_{a,pa(a)}, [a_1, x^*_{a,pa(a)}]^*, ..., [a_d, x^*_{a,pa(a)}]^*, \mathbf{x}_{pa(a)}, \mathbf{x}'_{R(a)}, \mathbf{x}_C)$$
$$\leq F(x^*_{a,pa(a)}, [a_1, x^*_{a,pa(a)}]^\epsilon, ..., [a_d, x^*_{a,pa(a)}]^\epsilon, \mathbf{x}_{pa(a)}, \mathbf{x}'_{R(a)}, \mathbf{x}_C) + 2\sum_{i=1}^{d} |T(a_i)|\epsilon$$

where the inequality is by invoking the induction hypothesis for each $[a_i, x^*_{a,pa(a)}]^*$. Similar reasoning as the base case gives us that this last expression is less than or equal to $F(x^{*,\epsilon}_{a,pa(a)}, [a_1, x^{*,\epsilon}_{a,pa(a)}]^\epsilon, ..., [a_d, x^{*,\epsilon}_{a,pa(a)}]^\epsilon, \mathbf{x}_{pa(a)}, \mathbf{x}'_{R(a)}, \mathbf{x}_C) + 2|T(a)|\epsilon$, completing the proof. □

Imagining a "super-root" r' such that $r' = pa(r)$ and $f_{\{r',r\}} \equiv 0$, selecting $a = r$, and selecting $\mathbf{x}_{r'}$ arbitrarily in Theorem 2.1, gives us an algorithm which will compute an $2|A|\epsilon$-optimal joint action, given access to F^ϵ. We shall refer to this algorithm as `BestAct`.

Theorem 2.2. *Suppose the action subgraph G_A is a tree. Given access to oracle $F^\epsilon(\cdot, \mathbf{x}_C)$, the* `BestAct` *algorithm computes a $2|A|\epsilon$-optimal joint action for joint context $\mathbf{x}_C$ in $O(m^2|E_A|)$ time.*

Proof. Fix an arbitrary variable a. Given $[a_i, \mathbf{x}_{pa(a_i)}]^\epsilon$ for each $a_i \in ch(a)$, $[a, \mathbf{x}_{pa(a)}]^\epsilon$ can be computed in $O(m)$ time by definition. Thus, computing $[r, \mathbf{x}_{r'}]^\epsilon$ can be done in $O(m^2|A|)$ time. □

Notice that any acyclic forest G_A can be handled by running the tree algorithm on each connected component. Suppose now that the action subgraph G_A is arbitrary, but admits a tree decomposition $\mathcal{T} = (\mathcal{A}, \mathcal{E})$ where $S \in \mathcal{A}$ is a subset of A. Let $w = \max_{S \in \mathcal{A}} |S|$, the treewidth of G_A.

Theorem 2.3. *Let $\mathcal{T} = (\mathcal{A}, \mathcal{E})$ be a tree decomposition of action subgraph G_A with treewidth w. Given access to oracle $F^\epsilon(\cdot, \mathbf{x}_C)$, the* `BestAct` *algorithm can be generalized to compute*

a $2|\mathcal{A}|\epsilon$*-optimal joint action for joint context* $\mathbf{x}_C$ *in* $O(m^{2w}|\mathcal{E}|)$ *time.*

Proof. The approach used when G_A is a tree can be generalized to run on the tree decomposition of an arbitrary action subgraph. Details are given in Section 2.8. □

Note that in order for the generalized version of `BestAct` to be computationally efficient, natural but nontrivial restrictions on the action subgraph G_A are required (namely, small treewidth). It can be shown that some restrictions are inevitable. For example, the energy F of a 3-D Ising model from statistical physics can be phrased as the sum of binary potential functions. However, even if the behavior of each potential function f_P is known (i.e. there is no implicit learning problem), and there are no contexts, it is still NP-hard to select the action (i.e. spins on the variables of the Ising graph) maximizing the energy function F Barahona (1982).

2.5. Payoff Estimator Subroutine

In this section we present the `PayEst` algorithm, which approximately implements the ϵ-good oracle $F^\epsilon(\mathbf{x}_A, \mathbf{x}_C)$ required by the `BestAct` algorithm described in Section 2.4. Before presenting `PayEst`, we give in Section 2.5.1 a precise definition of the problem that it is designed to solve. In Section 2.5.2 we give a simple solution for the special case where the observed payoffs are deterministic, which suffices to convey the main ideas of our approach. The solution to the general problem is given in Section 2.5.3, which will be an instance of a "Knows What It Knows" (KWIK) algorithm Li et al. (2011).

2.5.1. Problem Statement

Let us review the details of how the `BestAct` algorithm uses its oracle. On several time steps[1] $s = 1, \ldots, S$ `BestAct` specifies a complete joint assignment $\mathbf{x}^s = (\mathbf{x}^s_A, \mathbf{x}^s_C)$ and requests the value $F^\epsilon(\mathbf{x}^s)$ from the oracle. Ideally, we would like `PayEst` to unerringly supply these values to `BestAct`. However, since the true payoff function F is unknown, this will be

[1] These time steps are *not* the rounds of the bandit problem; `BestAct` may call the oracle several times per round of the bandit problem.

impossible in general. Instead, `PayEst` will be designed for the following learning protocol: On each time step s the algorithm must *either* output the value $\hat{f}^s = F^\epsilon(\mathbf{x}^s)$ *or* output a special symbol $\perp$ and be allowed to observe an independent random variable f^s with mean $F(\mathbf{x}^s)$. As we shall see in Section 2.6 when we integrate the two algorithms, `BestAct` and `PayEst` can be integrated in a way that respects this protocol and also bounds the number of times that `PayEst` outputs $\perp$.

2.5.2. Deterministic Payoffs Setting

Let us first consider the special case where each payoff f^s is in fact deterministic; that is, `PayEst` always observes $F(\mathbf{x}^s)$ directly. We give a simple algorithm for `PayEst` that is based on the idea that Assumption 2.1 allows us to express the payoff function F in a compact *linearized* form. We now proceed to describe this linearization, and then give the algorithm.

For convenience, let $N = \sum_{P \in \mathcal{P}} \prod_{i \in P} |X_i|$, and define the *payoff vector* $\mathbf{f} \in \mathbb{R}^N$ as follows: Divide the N components of $\mathbf{f}$ into $|\mathcal{P}|$ blocks, where each block corresponds to a potential function f_P. Within the block for potential function f_P, let there be one component ℓ corresponding to each of the $\prod_{i \in P} |X_i|$ possible joint assignments $\mathbf{x}_P$, and set the ℓth component of $\mathbf{f}$ equal to $f_P(\mathbf{x}_P)$.

For any complete joint assignment $\mathbf{x}$, let $\mathbf{v}(\mathbf{x}) \in \{0,1\}^N$ be a binary *coefficient vector* defined as follows: If ℓ is the component of $\mathbf{f}$ corresponding to potential function f_P and joint assignment $\mathbf{x}'_P$ then the ℓth component of $\mathbf{v}(\mathbf{x})$ equals 1 if and only if $\mathbf{x}_P = \mathbf{x}'_P$.

For an illustration of a payoff vector and coefficient vector, consider an expected payoff function F that depends on three variables and decomposes into two potential functions. Let $F(x_1, x_2, x_3) = f_{\{1,2\}}(x_1, x_2) + f_{\{2,3\}}(x_2, x_3)$, and suppose that the first potential adds its inputs, while the second potential multiplies them, i.e., $f_{\{1,2\}}(x_1, x_2) = x_1 + x_2$ and $f_{\{2,3\}}(x_2, x_3) = x_2 x_3$. If each variable x_i takes its values from the set $\{a, b\}$, then the payoff vector for the function F can be written $\mathbf{f} = \big(a+a, a+b, b+a, b+b, aa, ab, ba, bb\big)$ and the coefficient vector corresponding to, say, the complete joint assignment $\mathbf{x} = (x_1, x_2, x_3) =$

(a, b, a) is $\mathbf{v}(\mathbf{x}) = \big(0, 1, 0, 0, 0, 0, 1, 0\big)$. Most importantly, note that in general, the definitions of $\mathbf{f}$ and $\mathbf{v}(\mathbf{x})$, together with Assumption 2.1, imply that the expected payoff function has the linear form $F(\mathbf{x}) = \mathbf{f} \cdot \mathbf{v}(\mathbf{x})$.

Now a very natural `PayEst` algorithm presents itself: On each time step s, if there exists a linear combination $\alpha_1, \ldots, \alpha_{s-1}$ such that $\mathbf{v}(\mathbf{x}^s) = \sum_{s'=1}^{s-1} \alpha_{s'} \mathbf{v}(\mathbf{x}^{s'})$ — in other words, if $\mathbf{v}(\mathbf{x}^s)$ is in the linear span of previous coefficient vectors — then output the estimate $\hat{f}^s = \sum_{s'=1}^{s-1} \alpha_{s'} f^{s'}$, and otherwise output $\perp$. Clearly, because the expected payoff function $F(\mathbf{x})$ is a linear function of the coefficient vector $\mathbf{v}(\mathbf{x})$ and the observed payoffs are deterministic, we have $\hat{f}^s = F(\mathbf{x}^s)$ for all s. Also, since each coefficient vector is in $\mathbb{R}^N$, and there is no set of linearly independent vectors in $\mathbb{R}^N$ containing more than N vectors, the number of observation time steps is at most N. Importantly, we can upper bound N in terms of properties of the interaction graph G. Recall that $m = \max_i |X_i|$ and $n = |V|$, and suppose that each potential function $P \in \mathcal{P}$ is k-ary. We have $N = \sum_{P \in \mathcal{P}} \prod_{i \in P} |X_i| \leq |\mathcal{P}| m^k \leq \binom{n}{k} m^k \leq (mn)^k$. Thus, if we regard k as a constant, the number of observation time steps is upper bounded by a polynomial.

In fact, we can further exploit the structure of the interaction graph G to give a more refined upper bound than N. Let $M = \prod_{i \in V} |X_i|$, and define the *coefficient matrix* $\mathbf{M} \in \{0, 1\}^{N \times M}$ as follows: For each of the M possible complete joint assignments $\mathbf{x}$, the matrix $\mathbf{M}$ contains one column that equals the coefficient vector $\mathbf{v}(\mathbf{x})$. Since there is no set of linearly independent columns of $\mathbf{M}$ containing more than $\text{rank}(\mathbf{M})$ vectors, the number of observation time steps is at most $\text{rank}(\mathbf{M})$, which is at most N, but potentially much less than N. In Section 2.7, we give another result bounding $\text{rank}(\mathbf{M})$ in terms of properties of the interaction graph G.

2.5.3. Probabilistic Payoffs Setting

We now return to the general setting, so that each f^s is no longer deterministic, but an independent random variable with expected value $F(\mathbf{x}^s)$. Thanks to the linearized repre-

sentation of the expected payoff function F described in the previous section, we can use a *KWIK linear regression algorithm* Strehl and Littman (2007) to implement `PayEst`. On each time step $s = 1, \ldots, S$ such an algorithm observes a feature vector $\boldsymbol{\phi}^s$ and does exactly one of the following: (1) outputs prediction $\hat{y}^s$, or (2) outputs $\perp$ and observes independent random variable $y^s \in [0, 1]$ with expected value $\mathbf{w} \cdot \boldsymbol{\phi}^s$, where the weights $\mathbf{w}$ are unknown. In our case, $\hat{y}^s$ and y^s are the predicted and observed payoffs $\hat{f}^s$ and f^s, the feature vector $\boldsymbol{\phi}^s$ is the coefficient vector $\mathbf{v}(\mathbf{x}^s)$, and the unknown weight vector $\mathbf{w}$ is the payoff vector $\mathbf{f}$.

There are several existing algorithms for KWIK linear regression. For example, Cesa-Bianchi et al. (2009) describe an algorithm for which the number of observation time steps is upper bounded by $O\left(\frac{d}{\epsilon^2} \log(\frac{S}{\epsilon\delta})\right)$, where d is the dimension of the subspace containing the feature vectors. In our case, we have $d = \text{rank}(\mathbf{M})$, where $\mathbf{M}$ is the coefficient matrix. For concreteness, we will henceforth use this algorithm for `PayEst`.

2.6. Graphical Bandit Algorithm

In this section, we compose the algorithms from the previous two sections to form the `GraphicalBandit` algorithm, which is described in detail in Algorithm 1. In each round t, `GraphicalBandit` runs `BestAct` on the current joint context $\mathbf{x}_C^t$, and whenever `BestAct` asks the oracle for the value of $F^\epsilon(\mathbf{x}_A, \mathbf{x}_C^t)$ for some joint action $\mathbf{x}_A$, this request is passed on to `PayEst` as the coefficient vector $\mathbf{v}(\mathbf{x}_A, \mathbf{x}_C^t)$. If `PayEst` never returns $\perp$ for a given run of `BestAct`, then by the analysis in Sections 2.4 and 2.5, `BestAct` will return an ϵ-optimal joint action $\mathbf{x}_A^\epsilon$ for joint context $\mathbf{x}_C^t$, which is then played in round t. However, if `PayEst` returns $\perp$ in response to some complete joint assignment $(\mathbf{x}_A, \mathbf{x}_C^t)$, then `BestAct` is terminated immediately, the joint action $\mathbf{x}_A$ is played in round t, and the observed payoff f^t is returned to `PayEst`. Since `PayEst` is a KWIK linear regression algorithm, this feedback is required in order to bound the number of times that `PayEst` outputs $\perp$.

Theorem 2.4. *Let $d = \text{rank}(\mathbf{M})$ be the rank of the coefficient matrix, and let $\mathcal{T} = (\mathcal{A}, \mathcal{E})$ be the tree decomposition of the action subgraph G_A with treewidth w. The regret $R(T)$ of* `GraphicalBandit` *after T rounds is at most $R(T) \leq O\left(\frac{dw}{\epsilon^2} \log(Tm|\mathcal{E}|/\epsilon\delta) + 2|\mathcal{A}|\epsilon T + \delta T\right)$*

Algorithm 1 `GraphicalBandit`

Given: Subroutine `PayEst`, subroutine `BestAct`, parameters ϵ, δ.
Initialize `PayEst` with parameters ϵ, δ.
for each time step $t = 1, \ldots, T$ **do**
    Run `BestAct` on observed joint context $\mathbf{x}_C^t$.
    while `BestAct` is running **do**
        if `BestAct` asks for value of $F^\epsilon(\mathbf{x}_A, \mathbf{x}_C^t)$ **then**
            Input coeff. vector $\mathbf{v}(\mathbf{x}_A, \mathbf{x}_C^t)$ to `PayEst`.
            if `PayEst` returns $\perp$ **then**
                Play joint action $\mathbf{x}_A$, observe payoff f^t.
                Return f^t to `PayEst`.
                Terminate `BestAct` early.
            else if `PayEst` returns $\hat{f}$ **then**
                Return $\hat{f}$ to `BestAct`.
            end if
        end if
    end while
    if `BestAct` was not terminated early **then**
        Play joint action $\mathbf{x}_A^\epsilon$ returned by `BestAct`.
    end if
end for

and the computational complexity of each round is $O(m^{2w}|\mathcal{E}|)$.

Proof. The per round computational complexity follows from Theorem 2.3.

Since `PayEst` is the KWIK linear regression algorithm from Cesa-Bianchi et al. (2009), we have that with probability $1 - \delta$, every prediction $\hat{f}$ has error at most ϵ, and $\perp$ is returned at most $K \leq O\left(\frac{d}{\epsilon^2}\log(\frac{S}{\epsilon\delta})\right)$ times, where S is the number of times `BestAct` calls `PayEst`. Since `BestAct` call `PayEst` at most $O(m^{2w}|\mathcal{E}|)$ times per round, we have $S \leq O(Tm^{2w}|\mathcal{E}|)$. In the probability δ event that `PayEst` fails to meet its guarantee, we may suffer maximum regret on all T rounds. Otherwise, we are guaranteed that `BestAct` completes successfully on all but K rounds, and on the remaining rounds, by Theorem 2.3, a $2|\mathcal{A}|\epsilon$-optimal joint action is selected.

Thus, $R(T) \leq K + 2|\mathcal{A}|\epsilon T + \delta T \leq O\left(\frac{wd}{\epsilon^2}\log(Tm|\mathcal{E}|/\epsilon\delta) + 2|\mathcal{A}|\epsilon T + \delta T\right)$ □

If we tune ϵ and δ appropriately in terms of the number of rounds T, we obtain the following

no-regret bound.

Corollary 2.1. *If* `GraphicalBandit` *is run with parameters* $\epsilon = \delta = \frac{1}{T^{1/3}}$ *then* $R(T) \leq O\left(dw|\mathcal{A}|T^{2/3}\log(Tm|\mathcal{E}|)\right)$.

2.6.1. Distributional Assumptions

So far we have considered a setting in which contexts are chosen arbitrarily, and our regret bound in Theorem 2.4 depends on $d = \text{rank}(\mathbf{M})$. This is because, in the worst-case, each coefficient vector $\mathbf{v}(\mathbf{x}_A, \mathbf{x}_C)$ observed by `PayEst` will be linearly independent of all previously observed coefficient vectors. However, note that if the joint contexts are drawn from a distribution, then such a worst-case sequence may no longer be likely. Let $\mathcal{D}$ be an unknown *context distribution* on the joint contexts $\mathbf{x}_C$.

Fix a set $\mathbf{X}'_C \subseteq \mathbf{X}_C$, and let $\mathbf{M}(\neg\mathbf{X}'_C)$ be the coefficient matrix restricted to joint contexts in $\mathbf{X}_C \setminus \mathbf{X}'_C$. In other words, the columns of $\mathbf{M}(\neg\mathbf{X}'_C)$ are exactly $\{\mathbf{v}(\mathbf{x}_C, \mathbf{x}_A) \mid \mathbf{x}_A \in \mathbf{X}_A, \mathbf{x}_C \in \mathbf{X}_C \setminus \mathbf{X}'_C\}$. We can provide an alternate regret bound for the same algorithm by conceding full regret on rounds in which a context is chosen from $\mathbf{X}'_C$, and considering the algorithm's performance only on the remaining rounds. This gives us

$$R(T) \leq TP_{\mathbf{x}_C \sim \mathcal{D}}(\mathbf{x}_C \in \mathbf{X}'_C) + O\left(\text{rank}(\mathbf{M}(\neg\mathbf{X}'_C))w|\mathcal{A}|T^{2/3}\log(Tm|\mathcal{E}|)\right)$$

Thus if $\mathcal{D}$ places a large amount of its mass on joint contexts that generate a coefficient matrix with low rank, the bound above may be significantly better than our general bound. Note we can optimize this bound for rare contexts under the distribution $\mathcal{D}$ by minimizing this quantity over sets of joint contexts $\mathbf{X}'_C$.

2.7. Rank and Graph Structure

Let $\mathbf{M}_F$ be the coefficient matrix corresponding to a payoff function F, and $G_F = (V, E)$ be its interaction graph. In this section, we observe that we can use structural properties of G_F to prove statements about $\text{rank}(\mathbf{M}_F)$, and therefore the regret of our algorithm.

Consider a payoff function F that is the sum of binary potential functions (i.e., k-ary with $k = 2$). In Section 2.5 we argued that $\text{rank}(\mathbf{M}) = O(|V|^2)$. In this section we will argue that $\text{rank}(\mathbf{M}) = \Omega(|V|)$.

Theorem 2.5. *For any F with binary potentials,* $\text{rank}(\mathbf{M}_F) = \Omega(|V|)$.

Proof. Suppose that the largest matching in G_F contains at least $|V|/4$ edges. Since each potential function is binary, each edge in G_F represents a potential function in the decomposition of F, and so there is a subset $\mathcal{S} \subseteq \mathcal{P}$ of size at least $|V|/4$ such that all potentials $P \in \mathcal{S}$ are pairwise disjoint, and every $P \subseteq V$. Let $Y = \cup_{P \in \mathcal{S}} P$ be the set of nodes that participate in this matching.

For each such $P = \{a, a'\} \in \mathcal{S}$, fix an arbitrary "default" assignment $\mathbf{x}^0_P$ for the nodes $\{a, a'\}$. Let $P^- \triangleq Y \setminus P$, and furthermore, let $\mathbf{x}^0_{P^-}$ be the assignment in which all variables in P^- are set to their default assignments. For each $P = \{a, a'\} \in \mathcal{S}$, also let $\hat{X}_P = (X_a \times X_{a'}) \setminus \{\mathbf{x}^0_P\}$ be all possible assignments of P that are not the default assignment. Finally, let $\mathbf{x}'_{V \setminus Y}$ be an arbitrary joint assignment for the remaining variables.

For each $P \in \mathcal{S}$, and each each $\hat{\mathbf{x}}_P \in \hat{X}_P$, consider the complete joint assignments of the form $\mathit{complete}(\hat{\mathbf{x}}_P) = (\hat{\mathbf{x}}_P, \mathbf{x}^0_{P^-}, \mathbf{x}'_{V \setminus Y})$. It's not hard to see that the coefficient vectors corresponding these assignments are all linearly independent, since $\mathbf{v}(\mathit{complete}(\hat{\mathbf{x}}_P))$ is the only such coefficient vector that places a 1 in the component corresponding to the potential f_P and joint assignment $\hat{\mathbf{x}}_P$. Thus, $\mathbf{M}_F$ contains at least $|V|/4$ linearly independent columns, one for each of these complete assignments.

If the largest matching of G_F does not contain at least $|V|/4$ edges, then it contains at most $|V|/2$ nodes. Thus $B \triangleq V \setminus Y$ contains at least $|A|/2$ nodes. There cannot be an edge between two nodes in B, otherwise there would have been a larger matching. Thus for any $a \in B$, and $P \in \mathcal{P}$ with $a \in P$, we have $P \cap B = \{a\}$. Since G_F is an interaction graph, there must be at least one such P for each a, call it P_a. As in the previous case, we can construct a linearly independent set of at least $|V|/2$ coefficient vectors by fixing all vertices

to "default" values, considering each possible assignment of each variable $a \in B$, and finally considering the component of the coefficient vector corresponding to P_a and the particular assignment of a. □

Example 2.1. *There exists a class of functions $\mathcal{F}$ such that $\mathbf{M}_F$ has close to full rank for every $F \in \mathcal{F}$. In other words,* $\text{rank}(\mathbf{M}_F) = \Omega(N)$*, where $\mathbf{M}_F \in \{0,1\}^{N \times M}$ and $M > N$.*

Proof. Let $\mathcal{F}_n$ consist of functions F which are the sum of unary potentials, so $F = \sum_{i=1}^{n} f_i(\mathbf{x}_i)$. Let $\mathbf{X} = \mathbf{X}_1 \times ... \times \mathbf{X}_n$ where each $\mathbf{X}_i = \{0,1\}$. $M = 2^n$, while $N = 2n$. Let $\mathbf{x}^i$ be the joint assignment with $\mathbf{x}^i_i = 1$ and $\mathbf{x}^j_i = 0$ for all $j \neq i$. It's clear that $\{\mathbf{v}(\mathbf{x}^i_j)\}_j$ are linearly independent, establishing the claim. □

Example 2.2. *Any function F that does not decompose (i.e. with $\mathcal{P} = \{V\}$), has $\mathbf{M}_F$ equal to the identity matrix of size $|\mathbf{X}|$, and thus has* $\text{rank}(\mathbf{M}_F) = \prod_{i \in V} |X_i|$.

2.8. Extension to General Graphs

We prove a 'noise-free' version of Theorem 3.

Theorem 2.6. *Let $\mathcal{T} = (\mathcal{A}, \mathcal{E})$ be a tree decomposition of action subgraph G_A with treewidth w. Given access to oracle $F(\cdot, \mathbf{x}_C)$, the* `BestAct` *algorithm can be generalized to compute an optimal joint action for joint context $\mathbf{x}_C$ in $O(m^{2w}|\mathcal{E}|)$ time.*

As we explained in the case of trees, if we replace the oracle F with an ϵ-good oracle F^ϵ, we compute a $2|\mathcal{A}|\epsilon$-optimal joint action. This is because the ϵ noise 'accumulates', as shown in the proof of Theorem 1. However, carrying the ϵ noise through the proof given below is extremely cumbersome, and obscures the main ideas, so we only give a noise-free proof. To understand how the proof below can be extended to handle an ϵ-good oracle F^ϵ, see Theorem 1.

The generalization of `BestAct` that computes the optimal joint action in Theorem 3 is described in Algorithm 2 below. Before discussing the details of the algorithm, we present several key definitions and lemmas.

By the definition of tree decomposition, each node $S \in \mathcal{A}$ is a subset of action variables A, and $\mathcal{E}$ is a collection of edges over $\mathcal{A}$ that form a tree. We also know that $\mathcal{T}$ must satisfy the following properties:

1. *Node covering:* For every action variable $a \in A$ there exists a decomposition node $S \in \mathcal{A}$ such that $a \in S$.

2. *Edge covering:* For every edge $\{a, a'\} \in E$ there exists a decomposition node $S \in \mathcal{A}$ such that $a, a' \in S$.

3. *Running intersection:* If there exists an action variable $a \in A$ and decomposition nodes $S', S'' \in \mathcal{A}$ such that $a \in S'$ and $a \in S''$ then for every $S \in \mathcal{A}$ along the path between S' and S'' in tree $\mathcal{T}$ we have $a \in S$.

Root the tree $\mathcal{T}$ at an arbitrary decomposition node, and for any decomposition node $S \in \mathcal{A}$ define $\mathcal{T}(S)$ to be the subtree of $\mathcal{T}$ rooted at S. Also, define $\text{in}(S) \triangleq \bigcup_{S' \in \mathcal{T}(S)} S' \setminus S$ to be all the action variables contained in the decomposition nodes of subtree $\mathcal{T}(S)$, except for those contained in S. Finally, define $\text{out}(S) \triangleq \bigcup_{S' \notin \mathcal{T}(S)} S' \setminus S$ to be all the action variables contained in decomposition nodes not in subtree $\mathcal{T}(S)$, except for those contained in S.

Importantly, we chose the preceding definitions so that we can partition the set of action variables A in a convenient way, described in the next lemma.

Lemma 2.1. *For any decomposition node $S \in \mathcal{A}$ we have*[2]

$$A = S \sqcup \text{in}(S) \sqcup \text{out}(S)$$

Proof. The node covering property of tree decompositions implies that $A = S \cup \text{in}(S) \cup \text{out}(S)$. By definition, S is disjoint from $\text{in}(S)$ and $\text{out}(S)$, so it remains to show that $\text{in}(S)$ and $\text{out}(S)$ are disjoint. Suppose for contradiction that there exists an action variable $a \in A$ such that $a \in \text{in}(S)$ and $a \in \text{out}(S)$. Then there must exist $S' \in \mathcal{T}(S)$ and $S'' \notin \mathcal{T}(S)$ such

[2]Here and in the rest of this section, $\sqcup$ denotes disjoint union. In other words, $X = Y \sqcup Z$ is an assertion that $X = Y \cup Z$ and $Y \cap Z = \emptyset$.

that $a \in S'$ and $a \in S''$. Clearly, S is on the path in tree $\mathcal{T}$ from S' to S'', and therefore by the running intersection property of tree decompositions $a \in S$. But then by definition $a \notin \text{out}(S)$ and $a \notin \text{in}(S)$, a contradiction. □

Recall that $V = A \sqcup C$. Therefore, by Assumption 1 and Lemma 2.1, for any decomposition node $S \in \mathcal{A}$ we can write the payoff function F as

$$F(\mathbf{x}) = F(\mathbf{x}_S, \mathbf{x}_{\text{in}(S)}, \mathbf{x}_{\text{out}(S)}, \mathbf{x}_C) = \sum_{P \in \mathcal{P}} f_P(\mathbf{x}_{S \cap P}, \mathbf{x}_{\text{in}(S) \cap P}, \mathbf{x}_{\text{out}(S) \cap P}, \mathbf{x}_{C \cap P}).$$

The last expression can be simplified. The next lemma proves that, for any decomposition node S, there are two distinct categories of potential functions: those that depend on $\mathbf{x}_{\text{in}(S)}$, and those that depend on $\mathbf{x}_{\text{out}(S)}$.

Lemma 2.2. *For any decomposition node $S \in \mathcal{A}$ there exist collections of variables subsets $\mathcal{P}_{\text{in}}$, $\mathcal{P}_{\text{out}} \in 2^V$ such that $\mathcal{P} = \mathcal{P}_{\text{in}} \sqcup \mathcal{P}_{\text{out}}$, where $\mathcal{P}$ is the collection of variable subsets from Assumption 2.1, and*

$$F(\mathbf{x}) = \sum_{P \in \mathcal{P}_{\text{in}}} f_P(\mathbf{x}_{S \cap P}, \mathbf{x}_{\text{in}(S) \cap P}, \mathbf{x}_{C \cap P}) + \sum_{P \in \mathcal{P}_{\text{out}}} f_P(\mathbf{x}_{S \cap P}, \mathbf{x}_{\text{out}(S) \cap P}, \mathbf{x}_{C \cap P}).$$

Proof. It suffices to show that there is no subset of variables $P \in \mathcal{P}$ such that $P \cap \text{in}(S) \neq \emptyset$ and $P \cap \text{out}(S) \neq \emptyset$. Suppose for contradiction that such a subset $P \in \mathcal{P}$ exists. This implies that there exists action variables $a, a' \in A$ such that $a \in \text{in}(S)$, $a' \in \text{out}(S)$ and $a, a' \in P$. Since $a, a' \in P$, by our construction of graph $G = (A, E)$ we must have $\{a, a'\} \in E$. By the edge covering property of tree decompositions, there exists a tree decomposition node $S' \in \mathcal{T}$ such that $a, a' \in S'$. Clearly $S' \in \mathcal{T}(S)$ or $S' \notin \mathcal{T}(S)$. The former case implies that either $a' \in S$ or $a' \in \text{in}(S)$, and since $a' \in \text{out}(S)$, this contradicts the disjointness property given in Lemma 2.1. The latter case implies that either $a \in S$ or $a \in \text{out}(S)$, and since $a \in \text{in}(S)$, this also contradicts the disjointness property given in Lemma 2.1. □

We are now ready to present `BestAct`, an efficient algorithm for computing the best joint

action $\mathbf{x}_A^* = \arg\max_{\mathbf{x}_A} F(\mathbf{x}_A, \mathbf{x}_C)$ for a given joint context $\mathbf{x}_C$. Here is an informal description of `BestAct`: The algorithm processes the nodes of tree decomposition $\mathcal{T}$ one at a time, starting at the leaves and ending at the root. For each decomposition node $S \in \mathcal{A}$ and joint action $\mathbf{x}_S$, the algorithm computes $\text{best}(S, \mathbf{x}_S)$, which is the best assignment of values to action variables $\text{in}(S)$ when action variables S have values $\mathbf{x}_S$ and context variables C have values $\mathbf{x}_C$. Importantly, by Lemma 2.2, $\text{best}(S, \mathbf{x}_S)$ does *not* depend on the assignment of values to action variables $\text{out}(S)$. The value of $\text{best}(S, \mathbf{x}_S)$ is computed inductively: letting $S_1, \ldots, S_k$ be the children of S, the algorithm computes $\text{best}(S, \mathbf{x}_S)$ using previously computed values $\text{best}(S_1, \mathbf{x}'_{S_1}), \ldots, \text{best}(S_k, \mathbf{x}'_{S_k})$. After the algorithm reaches the root R of the tree decomposition, it computes the best joint action $\mathbf{x}_A^*$ using the values $\text{best}(R, \mathbf{x}_R)$.

When reading `BestAct`, and also the remainder of this section, it will be helpful to keep in mind that $S = (S \cap S') \sqcup (S \setminus S')$ for any sets S, S'.

Algorithm 2 `BestAct` for arbitrary action subgraphs

Given: Joint context $\mathbf{x}_C$, oracle access to $F(\cdot, \mathbf{x}_C)$, tree decomposition $\mathcal{T} = (\mathcal{A}, \mathcal{E})$ of action subgraph G_A.
for each node $S \in \mathcal{A}$ in postfix order (i.e., children before parents) **do**
  if S is a leaf **then**
    $\text{best}(S, \mathbf{x}_S) \leftarrow ()$ for each joint action $\mathbf{x}_S$; **continue**.
  end if
  for each joint action $\mathbf{x}_S$ **do**
    Let $S_1, \ldots, S_k$ be the children of S.
    for each child S_i of S **do**
      $\mathbf{x}^*_{S_i \setminus S} \leftarrow \arg\max_{\mathbf{x}'_{S_i \setminus S}} F\Big(\mathbf{x}_{S_i \cap S},\ \mathbf{x}'_{S_i \setminus S},\ \text{best}\left(S_i, \mathbf{x}_{S_i \cap S}, \mathbf{x}'_{S_i \setminus S}\right),\ \mathbf{x}'_{\text{out}(S_i)},\ \mathbf{x}_C\Big)$
      where joint action $\mathbf{x}'_{\text{out}(S_i)}$ can be chosen arbitrarily.
    end for
    $\text{best}(S, \mathbf{x}_S) \leftarrow \Big(\mathbf{x}^*_{S_1 \setminus S},\ \text{best}\left(S_1, \mathbf{x}_{S_1 \cap S}, \mathbf{x}^*_{S_1 \setminus S}\right),\ \ldots,\ \mathbf{x}^*_{S_k \setminus S},\ \text{best}\left(S_k, \mathbf{x}_{S_k \cap S}, \mathbf{x}^*_{S_k \setminus S}\right)\Big)$
  end for
end for
Let $R \in \mathcal{A}$ be root of decomposition $\mathcal{T}$.
$\mathbf{x}^*_R \leftarrow \arg\max_{\mathbf{x}'_R} F(\mathbf{x}'_R, \text{best}(R, \mathbf{x}'_R), \mathbf{x}_C)$. **return** $\mathbf{x}^*_A \leftarrow (\mathbf{x}^*_R, \text{best}(R, \mathbf{x}^*_R))$.

Before proceeding to the proof of Theorem 3, we prove a few more helpful lemmas. The next lemma gives a useful partition of $\text{in}(S)$.

Lemma 2.3. *For any decomposition node $S \in \mathcal{A}$ with children $S_1, \ldots, S_k \in \mathcal{A}$*

$$\text{in}(S) = (S_1 \setminus S) \sqcup \text{in}(S_1) \sqcup \cdots \sqcup (S_k \setminus S) \sqcup \text{in}(S_k)$$

Proof. For shorthand let $Z = \bigcup_{i=1}^{k} (S_i \setminus S) \cup \text{in}(S_i)$. First we prove $\text{in}(S) \subseteq Z$, then $\text{in}(S) \supseteq Z$, and finally that all pairs of sets of disjoint.

To show that $\text{in}(S) \subseteq Z$, consider an action variable $a \in \text{in}(S)$. By definition there exists $S' \in \mathcal{T}(S)$ such that $a \in S'$ and $a \notin S$. Therefore $S' \in \mathcal{T}(S_i)$ for some $i = 1, \ldots, k$. If $S' = S_i$ then clearly $a \in (S_i \setminus S)$. Otherwise if $a \notin S_i$ then $a \in \text{in}(S_i)$. Thus $\text{in}(S) \subseteq Z$.

To show that $\text{in}(S) \supseteq Z$, consider an action variable $a \in Z$. By definition there exists $i \in \{1, \ldots, k\}$ such that $a \in (S_i \setminus S)$ or $a \in \text{in}(S_i)$. The former case directly implies $a \in \text{in}(S)$. The latter case implies that there exists $S' \in \mathcal{T}(S_i)$ such that $a \in S'$. Now suppose for contradiction that $a \in S$. Since S_i is on the path from S and S' we have $a \in S_i$. But this contradicts the fact that $a \in \text{in}(S_i)$. Therefore $a \notin S$, which implies $a \in \text{in}(S)$.

Now we prove that all pairs of sets are disjoint. There are three types of pairs of sets: $(S_i \setminus S)$ and $\text{in}(S_j)$; $(S_i \setminus S)$ and $(S_j \setminus S)$; $\text{in}(S_i)$ and $\text{in}(S_j)$. We consider only the first case, since the proofs for the other two cases are similar. Suppose, for contradiction, that there exists an action variable $a \in A$ such that $a \in (S_i \setminus S)$ and $a \in \text{in}(S_j)$ for some $i, j \in \{1, \ldots, k\}$. Therefore $a \in S_i$ and there exists $S' \in \mathcal{T}(S_j)$ such that $a \in S'$. Since S is on the path between S_i and S', by the running intersection property we have $a \in S$, a contradiction. □

The next lemma proves the key property of the values $\text{best}(S, \mathbf{x}_S)$ that are computed in `BestAct`.

Lemma 2.4. *If* `BestAct` *is run on joint context $\mathbf{x}_C$ then after the outermost loop of the algorithm terminates we have for all decomposition nodes $S \in \mathcal{A}$ and all joint actions $\mathbf{x}_S$*

$$\text{best}(S, \mathbf{x}_S) \in \arg\max_{\mathbf{x}'_{\text{in}(S)}} F(\mathbf{x}_S, \mathbf{x}'_{\text{in}(S)}, \mathbf{x}'_{\text{out}(S)}, \mathbf{x}_C)$$

for any joint action $\mathbf{x}'_{\text{out}(S)}$.

Proof. The proof is by induction on the structure of the tree decomposition $\mathcal{T}$.

For the base case, S is a leaf of $\mathcal{T}$. The lemma clearly holds in this case, since by definition $\text{in}(S) = \emptyset$ and by line 4 of `BestAct` we have $\text{best}(S, \mathbf{x}_S) = ()$ for all joint actions $\mathbf{x}_S$.

Now suppose S is not a leaf of $\mathcal{T}$. Fix joint action $\mathbf{x}_S$. Let $S_1, \ldots, S_k$ be the children of S, and suppose for induction that the lemma holds for each child S_i. We wish to show

$$\text{best}(S, \mathbf{x}_S) \in \arg\max_{\mathbf{x}'_{\text{in}(S)}} F(\mathbf{x}_S, \mathbf{x}'_{\text{in}(S)}, \mathbf{x}'_{\text{out}(S)}, \mathbf{x}_C) \tag{2.1}$$

for any joint action $\mathbf{x}'_{\text{out}(S)}$. By line 12 of `BestAct` we have

$$\text{best}(S, \mathbf{x}_S) = \Big(\mathbf{x}^*_{S_1 \setminus S},\ \text{best}\left(S_1, \mathbf{x}_{S_1 \cap S}, \mathbf{x}^*_{S_1 \setminus S}\right),\ \ldots,\ \mathbf{x}^*_{S_k \setminus S},\ \text{best}\left(S_k, \mathbf{x}_{S_k \cap S}, \mathbf{x}^*_{S_k \setminus S}\right)\Big) \tag{2.2}$$

where $\mathbf{x}^*_{S_i \setminus S}$ for $i = 1, \ldots, k$ is defined in line 9 of `BestAct` as

$$\mathbf{x}^*_{S_i \setminus S} \in \arg\max_{\mathbf{x}'_{S_i \setminus S}} F\Big(\mathbf{x}_{S_i \cap S},\ \mathbf{x}'_{S_i \setminus S},\ \text{best}\left(S_i, \mathbf{x}_{S_i \cap S}, \mathbf{x}'_{S_i \setminus S}\right),\ \mathbf{x}'_{\text{out}(S_i)},\ \mathbf{x}_C\Big) \tag{2.3}$$

for some arbitrary joint action $\mathbf{x}'_{\text{out}(S_i)}$. By Lemma 2.3 we have

$$\text{in}(S) = (S_1 \setminus S) \sqcup \text{in}(S_1) \sqcup \cdots \sqcup (S_k \setminus S) \sqcup \text{in}(S_k). \tag{2.4}$$

Eq. (2.2) and Eq. (2.4) together imply that to show Eq. (2.1) it suffices to show

$$\begin{aligned}\Big(\mathbf{x}^*_{S_1 \setminus S},\ \text{best}\left(S_1, \mathbf{x}_{S_1 \cap S}, \mathbf{x}^*_{S_1 \setminus S}\right),\ \ldots,\ \mathbf{x}^*_{S_k \setminus S},\ \text{best}\left(S_k, \mathbf{x}_{S_k \cap S}, \mathbf{x}^*_{S_k \setminus S}\right)\Big)& \\ \in \arg\max_{\substack{\mathbf{x}'_{S_1 \setminus S}, \ldots, \mathbf{x}'_{S_k \setminus S} \\ \mathbf{x}'_{\text{in}(S_1)}, \ldots, \mathbf{x}'_{\text{in}(S_k)}}} F(\mathbf{x}_S, \mathbf{x}'_{S_1 \setminus S}, \mathbf{x}'_{\text{in}(S_1)}, \ldots, \mathbf{x}'_{S_k \setminus S}, \mathbf{x}'_{\text{in}(S_k)}, \mathbf{x}'_{\text{out}(S)}, \mathbf{x}_C)&\end{aligned} \tag{2.5}$$

for any joint action $\mathbf{x}'_{\text{out}(S)}$. Note that

$$\text{out}(S_i) = A \setminus \big(\text{in}(S_i) \sqcup S_i\big) = A \setminus \big(\text{in}(S_i) \sqcup (S_i \cap S) \sqcup (S_i \setminus S)\big)$$

for $i = 1, \ldots, k$, which follows from Lemma 2.1 and the fact that $S_i = (S_i \cap S) \sqcup (S_i \setminus S)$. Thus to show Eq. (2.5) it suffices to show for $i = 1, \ldots, k$

$$\Big(\mathbf{x}^*_{S_i \setminus S},\ \text{best}\big(S_i, \mathbf{x}_{S_i \cap S}, \mathbf{x}^*_{S_i \setminus S}\big)\Big) \in \arg\max_{\substack{\mathbf{x}'_{S_i \setminus S} \\ \mathbf{x}'_{\text{in}(S_i)}}} F\big(\mathbf{x}_{S_i \cap S}, \mathbf{x}'_{S_i \setminus S}, \mathbf{x}'_{\text{in}(S_i)}, \mathbf{x}'_{\text{out}(S_i)}, \mathbf{x}_C\big) \tag{2.6}$$

for any joint action $\mathbf{x}'_{\text{out}(S_i)}$. The inductive hypothesis implies that for $i = 1, \ldots, k$ and all joint actions $\mathbf{x}'_{S_i \setminus S}$

$$\text{best}\big(S_i, \mathbf{x}_{S_i \cap S}, \mathbf{x}'_{S_i \setminus S}\big) \in \arg\max_{\mathbf{x}'_{\text{in}(S_i)}} F\big(\mathbf{x}_{S_i \cap S}, \mathbf{x}'_{S_i \setminus S}, \mathbf{x}'_{\text{in}(S_i)}, \mathbf{x}'_{\text{out}(S_i)}, \mathbf{x}_C\big)$$

for any joint action $\mathbf{x}'_{\text{out}(S_i)}$. Thus to show Eq. (2.6) it suffices to show for $i = 1, \ldots, k$

$$\mathbf{x}^*_{S_i \setminus S} \in \arg\max_{\mathbf{x}'_{S_i \setminus S}} F\big(\mathbf{x}_{S_i \cap S}, \mathbf{x}'_{S_i \setminus S}, \text{best}\big(S_i, \mathbf{x}_{S_i \cap S}, \mathbf{x}'_{S_i \setminus S}\big), \mathbf{x}'_{\text{out}(S_i)}, \mathbf{x}_C\big) \tag{2.7}$$

for any joint action $\mathbf{x}'_{\text{out}(S_i)}$. By the definition of $\mathbf{x}^*_{S_i \setminus S}$ in Eq. (2.3) we have that Eq. (2.7) holds for at least one joint action $\mathbf{x}'_{\text{out}(S_i)}$. Together with Lemma 2.2 this implies that Eq. (2.7) holds for any joint action $\mathbf{x}'_{\text{out}(S_i)}$. □

We are now ready to prove Theorem 3.

Proof of Theorem 3. We first claim that the joint action $\mathbf{x}^*_A$ returned by `BestAct` satisfies

$$\mathbf{x}^*_A = \arg\max_{\mathbf{x}_A} F(\mathbf{x}_A, \mathbf{x}_C)$$

Let R be the root of the tree decomposition $\mathcal{T}$. By definition $\text{out}(R) = \emptyset$, and thus by

Lemma 2.1 we have $A = R \sqcup \text{in}(R)$. Therefore

$$\max_{\mathbf{x}_A} F(\mathbf{x}_A, \mathbf{x}_C) = \max_{\mathbf{x}_R, \mathbf{x}_{\text{in}(R)}} F(\mathbf{x}_R, \mathbf{x}_{\text{in}(R)}, \mathbf{x}_C) = \max_{\mathbf{x}_R} F(\mathbf{x}_R, \text{best}(R, \mathbf{x}_R), \mathbf{x}_C)$$

where the second equality follows from Lemma 2.4. By inspecting lines 15-17 of `BestAct`, the claim is proved.

Next, we claim that the running time of `BestAct` is $O(m^{2w}|\mathcal{E}|)$. By inspection, we see that the order of `BestAct`'s running time is determined by the time it spends in its innermost nested loop. Each iteration of the innermost loop corresponds to a different triple $(S, \mathbf{x}_S, S_i)$, where S and S_i are decomposition nodes connected by an edge in tree $\mathcal{T}$, and $\mathbf{x}_S$ is a joint action. There are $|\mathcal{E}|$ edges, and for any decomposition node S the number of joint actions $\mathbf{x}_S$ is at most m^w. Thus the innermost loop executes at most $m^w|\mathcal{E}|$ times. The body of the innermost loop consists solely of a maximization over all joint actions $\mathbf{x}_{S_i \setminus S}$; there are at most m^w such joint actions, yielding a total running time of $m^{2w}|\mathcal{E}|$. □

CHAPTER 3 : Large Scale Bandits and KWIK Learning

Continuing our examination of multi-armed bandit problems in which both the state and action spaces are very large, we study a more general model of similarity structure in the payoff function. In the previous chapter we were able to solve the graphical bandit problem by evoking a KWIK learning algorithm on a linearized form of the payoff function. We now observe that the connection between KWIK learning and the MAB problem in fact runs deeper.

Our main contribution is a new algorithm and reduction showing a strong connection between large-scale MAB problems and the *Knows What It Knows* or *KWIK* model of supervised learning Li et al. (2011); Li and Littman (2010); Sayedi et al. (2010); Strehl and Littman (2007); Walsh et al. (2009). KWIK learning is an online model of learning a class of functions that is strictly more demanding than standard no-regret online learning, in that the learning algorithm *must* either make an accurate prediction on each trial or output "don't know". The performance of a KWIK algorithm is measured by the number of such don't-know trials.

Our first results essentially show that the large-scale MAB problem given by a parametric class of payoff functions can be efficiently reduced to the supervised KWIK learning of the same class. Armed with existing algorithms for KWIK learning, such as for noisy linear regression Strehl and Littman (2007); Walsh et al. (2009), we thus obtain new algorithms for large-scale MAB problems. We also give a matching intractability result showing that the demand for KWIK learnability is necessary, in that it cannot be replaced with standard online no-regret supervised learning, or weaker models such as PAC learning, while still implying solution to the MAB problem. In this sense our reduction is tight with respect to its assumption of KWIK learning.

We then consider an alternative model in which the action space remains large, but in which only a subset is available to the algorithm at any time, and this subset is growing

with time. This even better models settings such as sponsored search, where the space of *possible* ads is very large, but at any moment the search engine can only display those ads that have actually been placed by advertisers. We again show that such MAB problems can be reduced to KWIK learning, provided the arrival rate of new actions is sublinear in the number of trials. We also give information-theoretic impossibility results showing that this reduction is tight, in that weakening its assumptions no longer implies solution to the MAB problem. We conclude with a brief experimental illustration of this arriving-action model.

The KWIK framework was designed to address model-free reinforcement learning in large-state finite-action MDPs (see Li and Littman (2010) in particular). Li, Littman, Walsh, and Strehl (2011) suggest that other machine learning problems, including bandit problems, could benefit from the perspective introduced by KWIK learning. We confirm this conjecture, and in more detail, demonstrate that $1/\epsilon$ and $1/\epsilon^2$ bounds for KWIK algorithms automatically imply regret rates of $\sqrt{T}$ and $T^{2/3}$ for MAB problems. These results demonstrate the importance of studying the KWIK model, not just because it is an interesting model in its own right, but for the purpose of understanding sequential decision-making problems more generally. Finally, we comment that our results also fall into the line of research showing reductions and relationships between bandit-style learning problems and traditional supervised learning models Langford and Zhang (2007); Beygelzimer et al. (2011); Beygelzimer and Langford (2009).

3.1. Large-Scale Multi-Armed Bandits (MAB)

The Setting. We consider a sequential decision problem in which a learner, on each round t, is presented with a *state* $\mathbf{x}^t$, chosen by Nature from a large state space $\mathcal{X}$. The learner responds by choosing an *action* $\mathbf{a}^t$ from a large action space $\mathcal{A}$. We assume that the learner's (noisy) payoff is $f_\theta(\mathbf{x}^t, \mathbf{a}^t) + \eta^t$, where η^t is i.i.d. with $\mathbb{E}[\eta^t] = 0$. The function f_θ is unknown to the learner, but is chosen from a (parameterized) family of functions $\mathcal{F}_\Theta = \{f_\theta : \mathcal{X} \times \mathcal{A} \to \mathbb{R}^+ \mid \theta \in \Theta\}$ that is known to the learner. We assume that every

$f_\theta \in \mathcal{F}_\Theta$ returns values bounded in $[0, 1]$. In general we make no assumptions on the sequence of states $\mathbf{x}^t$, stochastic or otherwise. An instance of such a MAB problem is fully specified by $(\mathcal{X}, \mathcal{A}, \mathcal{F}_\Theta)$.

We will informally use the term "large-scale MAB problem" to indicate that both $|\mathcal{X}|$ and $|\mathcal{A}|$ are large or infinite, and that we seek algorithms whose resource requirements are greatly sublinear or independent of both. This is in contrast to works in which either only $|\mathcal{X}|$ was assumed to be large Langford and Zhang (2007); Beygelzimer et al. (2011) (which we shall term "large-state"; it is also commonly called *contextual bandits* in the literature), or only $|\mathcal{A}|$ is large Kleinberg et al. (2008) (which we shall term "large-action"). We now define our notion of *regret*, which permits arbitrary sequences of states.

Definition 3.1. *An algorithm for the large-scale MAB problem* $(\mathcal{X}, \mathcal{A}, \mathcal{F}_\Theta)$ *is said to have* **no regret** *if, for any* $f_\theta \in \mathcal{F}_\Theta$ *and any sequence* $\mathbf{x}^1, \mathbf{x}^2, \ldots \mathbf{x}^T \in \mathcal{X}$*, the algorithm's action sequence* $\mathbf{a}^1, \mathbf{a}^2, \ldots \mathbf{a}^T \in \mathcal{A}$ *satisfies* $R_A(T)/T \to 0$ *as* $T \to \infty$*, where we define* $R(T) \triangleq E\left[\sum_{t=1}^T \max_{\mathbf{a}_*^t \in \mathcal{A}} f_\theta(\mathbf{x}^t, \mathbf{a}_*^t) - f_\theta(\mathbf{x}^t, \mathbf{a}^t)\right]$.

We shall be particularly interested in algorithms for which we can in fact provide fast *rates* of convergence to no regret.

Example: Pairwise Interaction Models. We introduce a running example we shall use to illustrate our assumptions and results; other examples are discussed later. Let states $\mathbf{x}$ and actions $\mathbf{a}$ both be (bounded norm) d-dimensional vectors of reals. Let θ be a (bounded) d^2-dimensional parameter vector, and let $f_\theta(\mathbf{x}, \mathbf{a}) = \sum_{1 \le i,j \le d} \theta_{i,j} x_i a_j$; we then define $\mathcal{F}_\Theta$ to be the class of all such models f_θ. In such models, the payoffs are determined by pairwise interactions between the variables, and both the sign and magnitude of the contribution of $x_i a_j$ is determined by the parameter $\theta_{i,j}$. For example, imagine an application in which each state $\mathbf{x}$ represents demographic and behavioral features of an individual web user, and each action $\mathbf{a}$ encodes properties of an advertisement that could be presented to the user. A zipcode feature in $\mathbf{x}$ indicating the user lives in an affluent neighborhood and a language feature in $\mathbf{a}$ indicating that the ad is for a premium housecleaning service might have a large

positive coefficient, while the same zipcode feature might have a large negative coefficient with a feature in **a** indicating that the service is not yet offered in the user's city.

3.2. Assumptions: KWIK Learnability and Fixed-State Optimization

We next articulate the two assumptions we shall require on the class $\mathcal{F}_\Theta$ in order to obtain resource-efficient no-regret MAB algorithms. The first is KWIK learnabilty of $\mathcal{F}_\Theta$, a strong notion of *supervised* learning, introduced by Li et al. in 2008 Li et al. (2008, 2011). The second is the ability to find an approximately optimal action for a *fixed* state. Either one of these conditions in isolation is clearly insufficient for solving the large-scale MAB problem: KWIK learning of $\mathcal{F}_\Theta$ has no notion of choosing actions, but instead assumes input-output pairs $\langle \mathbf{x}, \mathbf{a} \rangle, f_\theta(\mathbf{x}, \mathbf{a})$ are simply given; whereas the ability to optimize actions for fixed states is of no obvious value in our changing-state MAB model. We will show, however, that together these two assumptions can exactly compensate for each other's deficiencies and be combined to solve the large-scale MAB problem.

3.2.1. KWIK Learning

In the KWIK learning protocol Li et al. (2008), we assume we have an input space $\mathcal{Z}$ and an output space $\mathcal{Y} \subset \mathbb{R}$. The learning problem is specified by a function $f : \mathcal{Z} \to \mathcal{Y}$, drawn from a specified function class $\mathcal{F}$. The set $\mathcal{Z}$ can generally be arbitrary but, looking ahead, our reduction from a large-scale MAB problem $(\mathcal{X}, \mathcal{A}, \mathcal{F}_\Theta)$ to a KWIK problem will set the function class as $\mathcal{F} = \mathcal{F}_\Theta$ and the input space as $\mathcal{Z} = \mathcal{X} \times \mathcal{A}$, the joint state and action spaces.

The learner is presented with a sequence of observations $\mathbf{z}^1, \mathbf{z}^2, \ldots \in \mathcal{Z}$ and, immediately after observing $\mathbf{z}^t$, is asked to make a prediction of the value $f(\mathbf{z}^t)$, but is allowed to predict the value $\perp$ meaning "don't know". Specifically:

1: Nature selects $f \in \mathcal{F}$
2: **for** $t = 1, 2, 3, \ldots$ **do**
3: Nature selects $\mathbf{z}^t \in \mathcal{Z}$ and presents to learner

4: Learner predicts $y^t \in \mathcal{Y} \cup \{\perp\}$

5: **if** $y^t = \perp$ **then**

6: Learner observes value $f(\mathbf{z}^t) + \eta^t$,

7: where η^t is a bounded 0-mean noise term

8: **else if** $y^t \neq \perp$ and $|y^t - f(\mathbf{z}^t)| > \epsilon$ **then**

9: `FAIL` and **exit**

10: **end if**

11: `// Continue if` y^t `is` ϵ`-accurate`

12: **end for**

Thus in KWIK model the learner may confess ignorance on any trial. Upon a report of "don't know", where $y^t = \perp$, the learner is given feedback, receiving a noisy estimate of $f(\mathbf{z}^t)$. However, if the learner chooses to make a prediction of $f(\mathbf{z}^t)$, *no feedback* is received [1], and this prediction *must* be ϵ-accurate, or else the learner fails entirely. In the KWIK model the aim is to make only a *bounded number* of $\perp$ predictions, and thus make ϵ-accurate predictions on almost every trial.

Definition 3.2. *Let the error parameter be $\epsilon > 0$ and the failure parameter be $\delta > 0$. Then $\mathcal{F}$ is said to be* KWIK-learnable *with don't-know bound $\mathbf{B} = \mathbf{B}(\epsilon, \delta)$ if there exists an algorithm such that for any sequence $\mathbf{z}^1, \mathbf{z}^2, \mathbf{z}^3, \ldots \in \mathcal{Z}$, the sequence of predictions $y^1, y^2, \ldots \in \mathcal{Y} \cup \{\perp\}$ satisfies $\sum_{t=1}^{\infty} \mathbf{1}[y^t = \perp] \leq \mathbf{B}$, and the probability of `FAIL` is at most δ. Any class $\mathcal{F}$ is said to be* efficiently *KWIK-learnable if there exists an algorithm that satisfies the above condition and on every round runs in time* $\text{poly}(\epsilon^{-1}, \delta^{-1})$.

Example Revisited: Pairwise Interactions. We show that KWIK learnability holds here. Recalling that $f_\theta(\mathbf{x}, \mathbf{a}) = \sum_{1 \leq i,j \leq d} \theta_{i,j} x_i a_j$, we can linearize the model by viewing the KWIK inputs as having d^2 components $z_{i,j} = x_i a_j$, with coefficients $\theta_{i,j}$, and the KWIK learnability of $\mathcal{F}_\Theta$ simply reduces to KWIK noisy linear regression, which has an efficient algorithm Li et al. (2011); Strehl and Littman (2007); Walsh et al. (2009).

[1] This aspect of KWIK learning is crucial for our reduction.

3.2.2. Fixed-State Optimization

We next describe the aforementioned fixed-state optimization problem for $\mathcal{F}_\Theta$. Assume we have a fixed function $f_\theta \in \mathcal{F}_\Theta$, a fixed state $\mathbf{x} \in \mathcal{X}$, and some $\epsilon > 0$. Then an algorithm shall be referred to as a *fixed-state optimization* algorithm for $\mathcal{F}_\Theta$ if the algorithm makes a series of (action) queries $\mathbf{a}^1, \mathbf{a}^2, \ldots \in \mathcal{A}$, and in response to $\mathbf{a}^i$ receives approximate feedback y^i satisfying $|y^i - f_\theta(\mathbf{x}, \mathbf{a}^i)| \leq \epsilon$; and then outputs a final action $\hat{\mathbf{a}} \in \mathcal{A}$ satisfying $\arg\max_{\mathbf{a}\in\mathcal{A}}\{f_\theta(\mathbf{x}, \mathbf{a})\} - f_\theta(\mathbf{x}, \hat{\mathbf{a}}) \leq \epsilon$. In other words, for any fixed state $\mathbf{x}$, given access only to (approximate) input-output queries to $f_\theta(\mathbf{x}, \cdot)$, the algorithm quickly finds an (approximately) optimal action under f_θ and $\mathbf{x}$. It is not hard to show that if we define $\mathcal{F}_\Theta(\mathcal{X}, \cdot) = \{f_\theta(\mathbf{x}, \cdot) : \theta \in \Theta, \mathbf{x} \in \mathcal{X}\}$ — which defines a class of *large-action* MAB problems *induced* by the class $\mathcal{F}_\Theta$ of large-scale MAB problems, each one corresponding to a *fixed* state — then the assumption of fixed-state optimization for $\mathcal{F}_\Theta$ is in fact equivalent to having a no-regret algorithm for $\mathcal{F}_\Theta(\mathcal{X}, \cdot)$. In this sense, the reduction we will provide shortly can be viewed as showing that KWIK learnability bridges the gap between the large-scale problem $\mathcal{F}_\Theta$ and its induced large-action problem $\mathcal{F}_\Theta(\mathcal{X}, \cdot)$.

Example Revisited: Pairwise Interactions. We show that fixed-state optimization holds here. For any fixed state $\mathbf{x}$ we wish to approximately maximize the output of $f_\theta(\mathbf{x}, \mathbf{a}) = \sum_{i,j} \theta_{i,j} x_i a_j$ from approximate queries. Since $\mathbf{x}$ is fixed, we can view the coefficient on a_j as $\tau_j = \sum_i \theta_{i,j} x_i$. While there is no hope of distinguishing θ and $\mathbf{x}$, there is no need to: querying on the jth standard basis vector returns (an approximation to) the value of τ_j. After doing so for each dimension j, we can output whichever basis vector yielded the highest payoff.

3.3. A Reduction of MAB to KWIK

We now give a reduction and algorithm showing that the assumptions of both KWIK-learnability and fixed-state optimization of $\mathcal{F}_\Theta$ suffice to obtain an efficient no-regret algorithm for the MAB problem for $\mathcal{F}_\Theta$. The high-level idea of the algorithm is as follows. Upon

receiving the state $\mathbf{x}_t$, we attempt to simulate the assumed fixed-state optimization algorithm `FixedStateOpt` on $f_\theta(\mathbf{x}^t, \cdot)$. Unfortunately, we do not have the required oracle access to $f_\theta(\mathbf{x}^t, \cdot)$, due to the fact that the state changes with each action that we take. Therefore, we will instead make use of the assumed `KWIK` learning algorithm as a surrogate. So long as `KWIK` never outputs $\perp$, the optimization subroutine terminates with an approximate optimizer for $f_\theta(\mathbf{x}^t, \cdot)$. If `KWIK` returns $\perp$ sometime during the simulation of `FixedStateOpt`, we halt that optimization but increase the don't-know count of `KWIK`, which can only happen finitely often. The precise algorithm follows.

Algorithm 3 KWIKBandit: MAB Reduction to `KWIK` + `FixedStateOpt`

Initialize `KWIK` to learn unknown $f_\theta \in \mathcal{F}_\Theta$.
for $t = 1, 2, \ldots$ **do**
  $\mathbf{x}^t \xleftarrow{\text{receive}}$ MAB
  $i \xleftarrow{\text{set}} 0$
  `feedbackflag` $\xleftarrow{\text{set}}$ `FALSE`
  Init `FixedStateOpt`^{t} to optimize $f_\theta(\mathbf{x}^t, \cdot)$
  while $i \xleftarrow{\text{set}} i + 1$ **do**
    $\mathbf{a}_i^t \xleftarrow{\text{query}}$ `FixedStateOpt`^{t}
    if `FixedStateOpt`^{t} *terminates* **then**
      $\mathbf{a}^t \xleftarrow{\text{set}} \mathbf{a}_i^t$
      break while
    end if
    $\mathbf{z}^t = (\mathbf{x}^t, \mathbf{a}_i^t) \xrightarrow{\text{input}}$ `KWIK`
    $\hat{y}_i^t \xleftarrow{\text{output}}$ `KWIK`
    if $\hat{y}_i^t = \perp$ **then**
      $\mathbf{a}^t \xleftarrow{\text{set}} \mathbf{a}_i^t$
      `feedbackflag` $\xleftarrow{\text{set}}$ `TRUE`
      break while
    else
      $\hat{y}_i^t \xrightarrow{\text{feedback}}$ `FixedStateOpt`^{t}
    end if
  end while
  $\mathbf{a}^t \xrightarrow{\text{action}}$ MAB
  $f_\theta(\mathbf{x}^t, \mathbf{a}^t) + \eta^t = y^t \xleftarrow{\text{observe}}$ MAB
  if `feedbackflag` = `TRUE` **then**
    $y^t \xrightarrow{\text{feedback}}$ `KWIK`
  end if
end for

Theorem 3.1. *Assume we have a family of functions $\mathcal{F}_\Theta$, a KWIK-learning algorithm*

KWIK for $\mathcal{F}_\Theta$, and a fixed-state optimization algorithm `FixedStateOpt`*. Then the* average *regret of Algorithm 3, $R_A(T)/T$, will be arbitrarily small for appropriately-chosen ϵ and δ, and large enough T. Moreover, the running time is polynomial in the running time of* `KWIK` *and* `FixedStateOpt`*.*

Proof. We first bound the cost of Algorithm 3. Let us consider the result of one round of the outermost loop, i.e. for some fixed t. First, consider the event that `KWIK` does not `FAIL` on any trial, so we are guaranteed that $\hat{y}_i^t$ is an ϵ-accurate estimate of $f_\theta(\mathbf{x}^t, \mathbf{a}_i^t)$. In this case the **while** loop can be broken in one of two ways:

- `KWIK` returns $\perp$ on the pair $(\mathbf{x}^t, \mathbf{a}_i^t)$. In this case, because we have assumed a bounded range for f_θ, we can say that $\max_{\mathbf{a}_*^t} f_\theta(\mathbf{x}^t, \mathbf{a}_*^t) - f_\theta(\mathbf{x}^t, \mathbf{a}^t) \leq 1$.
- `FixedStateOpt` terminates and returns $\mathbf{a}^t$. But this $\mathbf{a}^t$ is ϵ-optimal per our definition, hence we have that $\max_{\mathbf{a}_*^t} f_\theta(\mathbf{x}^t, \mathbf{a}_*^t) - f_\theta(\mathbf{x}^t, \mathbf{a}^t) \leq \epsilon$.

Therefore, on a trial t, we can bound

$$\max_{\mathbf{a}_*^t} f_\theta(\mathbf{x}^t, \mathbf{a}_*^t) - f_\theta(\mathbf{x}^t, \mathbf{a}^t) \leq \mathbf{1}[\texttt{KWIK outputs } \perp \text{ on round } t] + \epsilon.$$

Taking the average over $t = 1, \ldots, T$ we have

$$\frac{1}{T} \sum_{t=1}^{T} \max_{\mathbf{a}_*^t} f_\theta(\mathbf{x}^t, \mathbf{a}_*^t) - f_\theta(\mathbf{x}^t, \mathbf{a}^t) \leq \frac{\mathbf{B}(\epsilon, \delta)}{T} + \epsilon \tag{3.1}$$

where $\mathbf{B}(\epsilon, \delta)$ is the don't-know bound of `KWIK`. Inequality (3.1) holds on the event that `KWIK` does not `FAIL`. By definition, the probability that it does `FAIL` is at most δ, and in that case all we can say is that $(1/T) \sum_{t=1}^{T} \max_{\mathbf{a}_*^t} f_\theta(\mathbf{x}^t, \mathbf{a}_*^t) - f_\theta(\mathbf{x}^t, \mathbf{a}^t) \leq 1$ Therefore, we have that:

$$\frac{R(T)}{T} \leq \frac{\mathbf{B}(\epsilon, \delta)}{T} + \epsilon + \delta \tag{3.2}$$

We must now show that the quantity on the right hand side of (3.2) vanishes with correctly chosen ϵ and δ. But this is achieved trivially: for any small $\gamma > 0$ if we select $\delta = \epsilon < \gamma/3$ and for $T > \frac{3\mathbf{B}(\epsilon,\delta)}{\gamma}$ we have that $\frac{\mathbf{B}(\epsilon,\delta)}{T} + \epsilon + \delta < \gamma$ as desired. □

Algorithm 3 is not exactly a no-regret MAB algorithm, since it requires parameter choices to obtain small regret. But this is easily remedied.

Corollary 3.1. *Under the assumptions of Theorem 3.1, there exists a no-regret algorithm for the MAB problem on $\mathcal{F}_\Theta$.*

Proof sketch. This follows as a direct consequence of Theorem 3.1 and a standard use of the "doubling trick" for selecting the input parameters in an online fashion. The simple construction runs a sequence of versions of Algorithm 3 with decaying choices of ϵ, δ. □

The interesting case occurs when $\mathcal{F}_\Theta$ is efficiently quick-learnable with a polynomial don't-know bound. In that case, we can obtain fast rates of convergence to no-regret. For all known KWIK algorithms $\mathbf{B}(\epsilon, \delta)$ is polynomial in ϵ^{-1} and poly-logarithmic in δ^{-1}. The following corollary is left as a straightforward exercise, following from equation (3.2).

Corollary 3.2. *If the don't-know bound of* `KWIK` *is $\mathbf{B}(\epsilon, \delta) = O(\epsilon^{-d} \log^k \delta^{-1})$ for some $d > 0, k \geq 0$ then we have*

$$R(T)/T = O\left(\left(\frac{1}{T}\right)^{\frac{1}{d+1}} \log^k T\right).$$

Example Revisited: Pairwise Interaction Models. As we have previously argued, the assumptions of KWIK learning and fixed-state optimization are met for the class of pairwise interaction models, so Theorem 3.1 can be applied directly, yielding a no-regret algorithm. More generally, a no-regret result can be obtained for any $\mathcal{F}_\Theta$ that can be similarly "linearized"; this includes the graphical models studied in Chapter 2 (where the main result can be viewed as a special case of Theorem 3.1). Other applications of Theorem 3.1 include $\mathcal{F}_\Theta$ that obey a Lipschitz condition, where we can apply covering techniques to obtain the

KWIK subroutine (details omitted), and various function classes in the boolean setting Li et al. (2011).

3.3.1. No Weaker General Reduction

While Theorem 3.1 provides general conditions under which large-scale MAB problems can be solved efficiently, the assumption of KWIK learnability of $\mathcal{F}_\Theta$ is still a strong one, with noisy linear regression being the richest problem for which there is a known KWIK algorithm. For this reason, it would be nice to replace the KWIK learning assumption with a weaker learning assumption [2]. However, in the following theorem, we prove (under standard cryptographic assumptions) that there is in fact *no* general reduction of the MAB problem for $\mathcal{F}_\Theta$ to a weaker model of supervised learning. More precisely, we show that the "next strongest" standard model of supervised learning after KWIK, which is no-regret on arbitrary sequences of trials, does not imply no-regret MAB. This immediately implies that even weaker learning models (such as PAC learnability) also cannot suffice for no-regret MAB.

Theorem 3.2. *There exists a class of models $\mathcal{F}_\Theta$ such that*

- *$\mathcal{F}_\Theta$ is fixed-state optimizable.*
- *There is an efficient algorithm A such that on an arbitrary sequence of T trials $\mathbf{z}^t$, A makes a prediction $\hat{y}^t$ of $y^t = f_\theta(\mathbf{z}^t)$ and receives y^t as feedback; and the total regret $err(T) \triangleq \sum_{t=1}^{T} |y^t - \hat{y}^t|$ is sublinear in T. Thus we have only no-regret supervised learning instead of the stronger KWIK learning.*
- *Under standard cryptographic assumptions, there is no polynomial-time algorithm for the no-regret MAB problem for $\mathcal{F}_\Theta$, even if the state sequence is generated randomly from the uniform distribution.*

We sketch the proof of Theorem 3.2 in the remainder of the section. Let $\mathbb{Z}_n = \{0, ..., n-1\}$

[2] Note that we should not expect to replace or weaken the assumption of fixed-state optimization, since we have already noted that this is already implied by a no-regret algorithm for the MAB problem.

with $d = \log |\mathbb{Z}_n|$. The idea is that for each $f_\theta \in \mathcal{F}_\Theta$, the parameters θ specify the public-private key pair in a family of trapdoor functions $h_\theta : \mathbb{Z}_n \to \mathbb{Z}_n$ — thus informally, h_θ is computationally easy to compute, but computationally intractable to invert, unless one knows the private key, in which case it becomes easy. We then define the MAB functions f_θ as follows: for any state $\mathbf{x} \in \mathbb{Z}_n$, $f_\theta(\mathbf{x}, \mathbf{a}) = 1$ if $\mathbf{x} = h_\theta(\mathbf{a})$ and $f_\theta(\mathbf{x}, \mathbf{a}) = 0$ otherwise. Thus for any fixed state $\mathbf{x}$, finding the optimal action requires inverting the trapdoor function without knowledge of the private key, which is assumed intractable. In order to ensure fixed-state optimizability, we also introduce a "special" input $\mathbf{a}^*$ such that the value of $f_\theta(\mathbf{x}, \mathbf{a}^*)$ "gives away" the optimal action $h_\theta^{-1}(\mathbf{x})$ but with low payoff, but a MAB algorithm cannot exploit this since executing $\mathbf{a}^*$ in the environment changes the state.

Let $h_\theta^{-1}(\cdot)$ denote the inverse of h_θ. That is, $h^{-1}(\mathbf{x})$ is the optimizing action for state $\mathbf{x}$. If $\mathbf{a}^*$ is selected, the output of $f_\theta(\mathbf{x}, \mathbf{a}^*)$ is equal to $0.5/(1 + h_\theta^{-1}(\mathbf{x}))$. [3] Note that querying $\mathbf{a}^*$ in state $\mathbf{x}$ reveals the identity of $h_\theta^{-1}(\mathbf{x}) \in \mathbb{Z}_n$, but has vanishing payoff. Suppose also that the public key, θ_{pub}, is revealed as side information on any input to h_θ.[4]

The following lemma establishes that the previous construction admits trivial algorithms for both the fixed-state optimization problem and the no-regret supervised learning problem.

Lemma 3.1. *Let $\mathcal{F}_\Theta$ be the function class just described. For any $f_\theta \in \mathcal{F}_\Theta$ and any fixed $\mathbf{x} \in \mathcal{X}$, $f(\mathbf{x}, \cdot)$ can be optimized from a constant number of queries. Furthermore, there exists an efficient algorithm for the supervised no-regret problem on $\mathcal{F}_\Theta$ with $err(T) = O(\log T)$, and requiring $poly(d)$ computation per step.*

Proof. For any θ, the fixed-state optimization problem on $f_\theta(\mathbf{x}, \cdot)$ is solved by simply querying the special action $\mathbf{a}^*$, which uniquely identifies the optimal action. The supervised no-regret problem is similarly trivial. After the first observation $(\mathbf{x}^1, \mathbf{a}^1)$, θ_{pub} is revealed. Thereafter, the algorithm has learned the output of every pair $(\mathbf{x}, \mathbf{a})$, where both $\mathbf{x}$ and $\mathbf{a}$

[3] For simplicity think of each h_θ as being a bijection (H_θ is a family of one-way permutations).In general h_θ need not be a bijection if we let $h_\theta^{-1}(\mathbf{x})$ be an arbitrary inversion if many exist, and let $f_\theta(\mathbf{x}, \mathbf{a}^*) = 0$ if there is no $\mathbf{a} \in \mathbb{Z}_n$ satisfying $h_\theta(\mathbf{a}) = \mathbf{x}$.

[4] Once again, this keeps the construction simple. For complete rigor, the identity of θ_{pub} can be encoded in $O(d)$ bits and output after the lowest-order bit used by the described construction.

belong to $\mathbb{Z}_n$. (It simply checks if $h_\theta(\mathbf{a}) = \mathbf{x}$). The only inputs on which it might make a mistake take the form $(\mathbf{x}, \mathbf{a}^*)$. However, repeating the output for previously observed inputs, and outputting 0 for new inputs of the form $(\mathbf{x}, \mathbf{a}^*)$ suffices to solve the supervised no-regret problem with $\text{err}(T) = O(\log T)$. The algorithm cannot suffer error greater than $\sum_{t=1}^{T} 0.5/t$ in this way. □

Finally, we can demonstrate that an efficient no-regret algorithm for the large-scale bandit problem on $\mathcal{F}_\Theta$ gives us an algorithm for inverting h_θ.

Lemma 3.2. *Under standard cryptographic assumptions, there is no polynomial q and efficient algorithm* `BANDIT` *for the large-scale bandit problem on $\mathcal{F}_\Theta$ that guarantees*

$$\sum_{t=1}^{T} \max_{\mathbf{a}_*^t} f_\theta(\mathbf{x}_t, \mathbf{a}_*^t) - f_\theta(\mathbf{x}_t, \mathbf{a}^t) < .5T$$

with probability greater than $1/2$ when $T \leq q(d)$.

Proof. Suppose that there were such a q, and algorithm `BANDIT`. This would imply that h_θ can be inverted for arbitrary θ, while only knowing the public key θ_{pub}.

Consider the following procedure that simulates `BANDIT` for $q(d)$ steps. On each round t, the state provided to `BANDIT` will be generated by selecting an action $\mathbf{a}^t$ from $\mathbb{Z}_n$ uniformly at random, and then providing `BANDIT` with the state $h_\theta(\mathbf{a}^t)$. At which point, `BANDIT` will output an action and demand a reward. If the action selected by bandit is the special action $\mathbf{a}^*$, then its reward is simply $0.5/(1 + \mathbf{a}^t)$. If the action selected by bandit is $\mathbf{a}^t$ its reward is 1. Otherwise, it's reward is 0.

With probability $1/2$, `BANDIT` must return $\mathbf{a}^t$ on state $h_\theta(\mathbf{a}^t)$ for at least one round $t \leq T$. Before being able to invert $h_\theta(\mathbf{a}^t)$, the procedure described reveals at most $q(d)$ plaintext-ciphertext pairs $(\mathbf{a}^s, h_\theta(\mathbf{a}^s)), s < t$ to `BANDIT` (and no additional information), contradicting the assumption that h_θ belongs to a family of cryptographic trapdoor functions. □

This completes the proof of Theorem 3.2.

3.4. A Model for Gradually Arriving Actions

In the model examined so far, we have been assuming that the action space $\mathcal{A}$ is large — exponentially large or perhaps infinite — but also that the *entire* action space is available on every trial. In many natural settings, however, this property may be violated. For instance, in sponsored search, while the space of all possible ads is indeed very large, at any given moment the search engine can choose to display only those ads that have actually been created by extant advertisers. Furthermore these advertisers arrive gradually over time, creating a growing action space. In this setting, the algorithm of Theorem 3.1 cannot be applied, as it assumes the ability to optimize over *all* of $\mathcal{A}$ at each step. In this section we introduce a new model and algorithm to capture such scenarios.

Setting. As before, the learner is presented with a sequence of arriving states $\mathbf{x}^1, \mathbf{x}^2, \mathbf{x}^3, \ldots \in \mathcal{X}$. The set of available actions, however, shall not be fixed in advance but instead will grow with time. Let $\mathcal{F}$ be the set of all possible actions where, formally, we shall imagine that each $f \in \mathcal{F}$ is a function $f : \mathcal{X} \to [0, 1]$; $f(\mathbf{x})$ represents the payoff of action f on $\mathbf{x} \in \mathcal{X}$ [5]. Initially the action pool is $\mathcal{F}^0 \subset \mathcal{F}$, and on each round t a (possibly empty) set of new actions $S^t \subset \mathcal{F}$ arrives and is added to the pool, hence the available action pool on round t is $\mathcal{F}^t := \mathcal{F}^{t-1} \cup S^t$. We emphasize that when we say a new set of actions "arrives", we do *not* mean that the learner is given the actual *identity* of the corresponding functions, which it must learn to approximate, but rather that the learner is given (noisy) black-box input-output access to them. Let $N(t) = |\mathcal{F}^t|$ denote the size of the action pool at time t. Our results will depend crucially on this growth rate $N(t)$, in particular on it being *sublinear* [6]. One interpretation of this requirement, and our theorem that exploits it, is as a

[5] Note that now each action is represented by its own payoff function, in contrast to the earlier model in which actions were inputs $\mathbf{a}$ into the $f_\theta(\mathbf{x}, \mathbf{a})$. The models coincide if we choose $\mathcal{F} = \{f_\theta(\cdot, \mathbf{a}) : \mathbf{a} \in \mathcal{A}, \theta \in \Theta\}$.

[6] Sublinearity of $N(t)$ seems a mild and natural assumption in many settings; certainly in sponsored search we expect user queries to vastly outnumber new advertisers. Another example is crowdsourcing systems, where the arriving actions are workers that can be assigned tasks, and $f(\mathbf{x})$ is the quality of work that worker f does on task $\mathbf{x}$. If the workers are also the contributors of tasks (as in services like stackoverflow.com), and each worker contributes tasks at some constant rate, it is easily verified that $N(t) = \sqrt{t}$.

form of Occam's Razor: since new functions arriving means more parameters for the MAB algorithm to learn, it turns out to be necessary and sufficient that they arrive at a strictly slower rate than the data (trials).

We now precisely state the *arriving action* learning protocol:

1: Learner given an initial action pool $\mathcal{F}^0 \subset \mathcal{F}$
2: **for** $t = 1, 2, 3, \ldots$ **do**
3: Learner receives new actions $S^t \subset \mathcal{F}$ and updates pool $\mathcal{F}^t \leftarrow \mathcal{F}^{t-1} \cup S^t$
4: Nature selects state $\mathbf{x}^t \in \mathcal{Z}$ and presents to learner
5: Learner selects some $f^t \in \mathcal{F}^t$, and receives payoff $f^t(\mathbf{x}^t)+\eta^t$; η^t is i.i.d. with $\mathbb{E}[\eta^t] = 0$
6: **end for**

We now define our notion of regret for the arriving action protocol.

Definition 3.3. *Let A be an algorithm for making a sequence of decisions $f^1, f^2, \ldots$ according to the arriving action protocol. Then we say that A has* no regret *if on any sequence of pairs $(S^1, \mathbf{x}^1), (S^2, \mathbf{x}^2), \ldots, (S^t, \mathbf{x}^T)$, $R_A(T)/T \to 0$ as $T \to \infty$, where we re-define $R_A(T) \triangleq E\left[\sum_{t=1}^T \max_{f_*^t \in \mathcal{F}^t} f_*^t(\mathbf{x}^t) - \sum_{t=1}^T f^t(\mathbf{x}^t)\right]$.*

Reduction to KWIK Learning. Similar to Section 3.3, we now show how to use the KWIK learnability assumption on $\mathcal{F}$ to construct a no-regret algorithm in the arriving action model. The key idea, described in the reduction below, is to endow each action f in the current action pool with its own $\texttt{KWIK}_f$ subroutine. On every round, after observing the task $\mathbf{x}^t$, we shall query $\texttt{KWIK}_f$ for a prediction of $f(\mathbf{x}^t)$ for each $f \in W^t$. If *any* subroutine $\texttt{KWIK}_f$ returns $\perp$, we immediately stop and play action $f^t \leftarrow f$. This can be thought of as an *exploration step* of the algorithm. If every $\texttt{KWIK}_f$ returns a value, we simply choose the arg max as our selected action.

Theorem 3.3. *Let A denote Algorithm 4. For any $\epsilon > 0$ and any choice of $\{\mathbf{x}^t, S^t\}$,*

$$R_A(T) \leq N(T)\mathbf{B}(\epsilon, \delta) + 2T\epsilon + \delta N(T)T$$

Algorithm 4 No-Regret Learning in the Arriving Action Model

for $t = 1, 2, 3, \ldots$ **do**
  Learner receives new actions S^t
  Learner observes task $\mathbf{x}^t$
  for $f \in S^t$ **do**
    Initialize a subroutine $\texttt{KWIK}_f$ for learning f
  end for
  for $f \in \mathcal{F}^t$ **do**
    Query $\texttt{KWIK}_f$ for prediction $\hat{y}_f^t$
    if $\hat{y}_f^t = \perp$ **then**
      Take action $f^t = f$
      Observe $y^t \leftarrow f^t(\mathbf{x}^t)$
      Input y^t into $\texttt{KWIK}_f$, and **break**
    end if
  end for
  `// If no KWIK subroutine`
  `// returns ⊥, simply choose best!`
  Take action $f^t = \arg\max_{f \in \mathcal{F}^t} \hat{y}_f^t$
end for

where $\mathbf{B}(\epsilon, \delta)$ *is a bound on the number of* $\perp$ *returned by the KWIK-Subroutine used in A.*

Proof. The probability that at least one of the $N(T)$ `KWIK` algorithms will `FAIL` is at most $\delta N(T)$. In that case, we suffer the maximum possible T regret, accounting for the $\delta N(T)T$ term. Otherwise, on each round t we query every $f \in \mathcal{F}^t$ for a prediction, and either one of two things can occur: (a) $\texttt{KWIK}_f$ reports $\perp$ in which case we can suffer regret at most 1; or (b) each $\texttt{KWIK}_f$ returns a real prediction $\hat{y}_f^t \neq \perp$ that is ϵ-accurate, in which case we are guaranteed that the regret of f^t is no more than 2ϵ. More precisely, we can bound the regret on round t as

$$\max_{f_*^t \in \mathcal{F}^t} f_*^t(\mathbf{x}^t) - f^t(\mathbf{x}^t) \leq \mathbf{1}[\texttt{KWIK}_f \text{ outputs } \hat{y}_f^t = \perp \text{ for some } f] + 2\epsilon.$$

Of course, the total number of times that any $\texttt{KWIK}_f$ subroutine returns $\perp$ is no more than $\mathbf{B}(\epsilon, \delta)$, hence the total number of $\perp$'s after T rounds is no more than $N(T)\mathbf{B}(\epsilon, \delta)$. Summing (3.4) over $t = 1, \ldots, T$ gives the desired bound and we are done. $\square$

As a consequence of the previous theorem, we achieve a simple corollary:

Corollary 3.3. *Assume that* $\mathbf{B}(\epsilon, \delta) = O(\epsilon^{-d} \log^k \delta^{-1})$ *for some* $d > 0$, *and* $k \geq 0$. *Then*

$$\frac{R_A(T)}{T} = O\left(\left(\frac{N(T)}{T}\right)^{1/(d+1)} \log^k T\right)$$

which tends to 0 as long as $N(T)$ *is "slightly" sublinear in* T; $T = o(T/\log^{k(d+1)}(T))$.

Proof. Without loss of generality we can assume $\mathbf{B}(\epsilon, \delta) \leq \frac{c}{\epsilon}^{-d} \log \delta^{-1}$ for all ϵ, δ and some constant $c > 0$. Applying Theorem 3.3 gives:

$$\frac{R_A(T)}{T} \leq \frac{N(T)}{T} \frac{c}{\epsilon^d} + 2\epsilon + \delta N(T)$$

Choosing $\delta = 1/T$ and $\epsilon = \left(\frac{N(T)}{T}\right)^{1/(d+1)}$ allows us to conclude that

$$R_A(T)/T \leq (c+2)\left(\frac{N(T)}{T}\right)^{1/(d+1)} \log^k T$$

and hence we are done. □

Impossibility Results. The following two theorems show that our assumptions of the KWIK learnability of $\mathcal{F}$ and sublinearity of $N(t)$ are both necessary, in the sense that relaxing either is not sufficient to imply a no-regret algorithm for the arriving action MAB problem. Unlike the corresponding impossibility result of Theorem 3.2, the ones below do not rely on any complexity-theoretic assumptions, but are information-theoretic.

Theorem 3.4. *(Relaxing sublinearity of* $N(t)$ *insufficient to imply no-regret on MAB) There exists a class* $\mathcal{F}$ *that is KWIK-learnable with a don't-know bound of 1 such that if* $N(t) = t$, *for any learning algorithm* A *and any* T, *there is a sequence of trials in the arriving action model such that* $R_A(T)/T > c$ *for some constant* $c > 0$.

The full proof of Theorem 3.4 is provided in Section 3.6.1. The idea is to construct a class of T different functions in which the observation of a single labeled example *exactly* identifies

the function, thus rendering the class trivially KWIK-learnable; but under a linear arrival rate of such functions, a MAB algorithm can never "catch up" and suffers a constant regret rate.

Theorem 3.5. *(Relaxing KWIK to supervised no-regret insufficient to imply no-regret on MAB) There exists a class $\mathcal{F}$ that is supervised no-regret learnable such that if $N(t) = \sqrt{t}$, for any learning algorithm A and any T, there is a sequence of trials in the arriving action model such that $R_A(T)/T > c$ for some constant $c > 0$.*

Proof. (Sketch) First we describe the class $\mathcal{F}$. For any n-bit string x, let f_x be a function such that $f_x(x)$ is some large value, and for any $x' \neq x$, $f_x(x') = 0$. It's easy to see that $\mathcal{F}$ is not KWIK learnable with a polynomial number of don't-knows — we can keep feeding an algorithm different inputs $x' \neq x$, and as soon as the algorithm makes a prediction, we can reselect the target function to force a mistake (this is essentially the same argument proving that monomials are not KWIK learnable). $\mathcal{F}$ is no-regret learnable, however: we just keep predicting 0. As soon as we make a mistake, we learn x, and we'll never err again, so our regret is at most $O(1/T)$.

Now in the arriving action model, suppose we initially start with r distinct functions/actions $f_i = f_{x_i} \in \mathcal{F}$, $i = 1, \ldots, r$. We will choose $N(T) = \sqrt{T}$, which is sublinear, and $r = \sqrt{T}$, and we can make T as large as we want. So we have a no-regret-learnable $\mathcal{F}$ and a sublinear arrival rate; now we argue that the arriving action MAB problem is hard.

Pick a random permutation of the f_i, and let i be the indices in that order for convenience. We start the task sequence with all x_1's. The MAB learner faces the problem of figuring out which of the unknown f_is has x_1 as its high-payoff input. Since the permutation was random, the expected number of assignments of x_1 to different f_i before this is learned is $r/2$. At that point, all the learner has learned is the identify of f_1 — the fact that it learned that other $f_i(x_1) = 0$ is subsumed by learning $f_1(x_1)$ is large, since the f_i are all distinct.

We then continue the sequence with x_2's until the MAB learner identifies f_2, which now takes

$(r-1)/2$ assignments in expectation. Continuing in this vein, the number of assignments made before learning (say) half of the f_i is $\sum_{j=1}^{r/2}(r-j)/2 \approx \Omega(r^2) \approx \Omega(T)$. On this sequence of $\Omega(T)$ tasks, the MAB learner will have gotten non-zero payoff on only $r = \sqrt{T}$. The offline optimal, on the other hand, always knows the identity of the f_i and gets large payoff on every single task. So any learner's cumulative regret to offline grows linearly with T. $\square$

3.5. Experiments

We now give a brief experimental illustration of our models and results. For the sake of brevity we examine only our algorithm in the arriving action model just discussed. We consider a setting in which both states $\mathbf{x}$ and the actions or functions $\mathbf{f}$ are described by unit-norm, 10-dimensional real vectors, and the value taking $\mathbf{f}$ in state $\mathbf{x}$ is simply the inner product $\mathbf{f} \cdot \mathbf{x}$. For this class of functions we thus implemented the KWIK linear regression algorithm Walsh et al. (2009), which is given a fixed accuracy target or threshold of $\epsilon = 0.1$, and which is simulated with Gaussian noise added to payoffs with $\sigma = 0.1$. New actions/functions arrived stochastically, with the probability of a new $\mathbf{f}$ being added on trial t being $0.1/\sqrt{t}$; thus in expectation we have sublinear $N(t) = O(\sqrt{t})$. Both the $\mathbf{x}$ and the $\mathbf{f}$ are selected uniformly at random. On top of the KWIK subroutine, we implemented Algorithm 4.

In Figure 1 we show snapshots of simulations of this algorithm at three different timescales — after 1000, 5000, and 25,000 trials respectively. The snapshots are indeed from three independent simulations in order to illustrate the variety of behaviors induced by the exogenous stochastic arrivals of new actions/functions, but also to show typical performance for each timescale.

In each subplot, we plot three quantities. The blue curve show the average reward per step so far for the omniscient offline optimal that is *given* each weight $\mathbf{f}$ as it arrives, and thus always chooses the optimal available action on every trial. This curve is the best possible performance, and is the target of the learning algorithm. The red curve shows the average

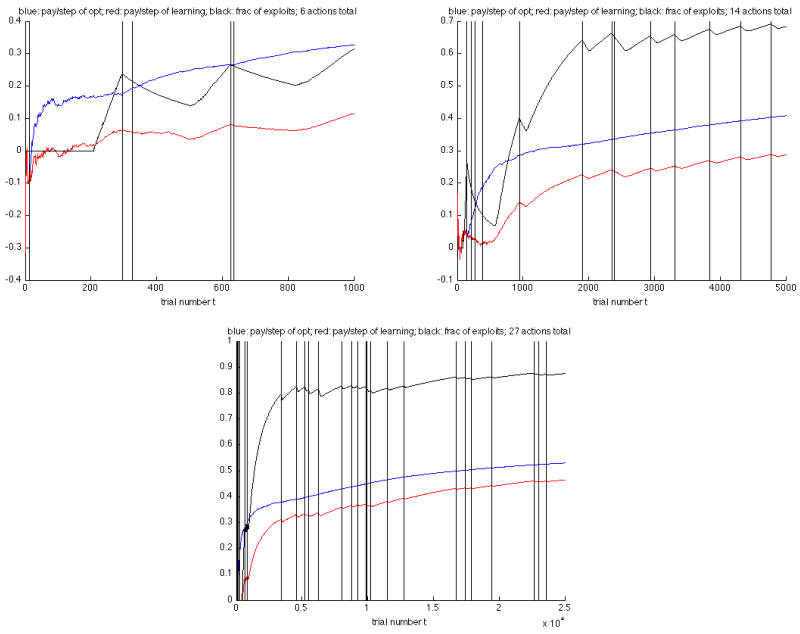

Figure 1: Simulations of Algorithm 4 at three timescales; see text for details.

reward per step so far for Algorithm 4. The black curve shows the fraction of *exploitation steps* for the algorithm so far (the last line of Algorithm 4, where we are guaranteed to choose an approximately optimal action). The vertical lines indicate trials in which a new action/function was added.

First considering $T = 1000$ (left panel, in which a total of 6 actions are added), we see that very early (as soon as the second action arrives, and thus there is a choice over which the offline omniscient can optimize) the algorithm badly underperforms, and is never exploiting — new actions are arriving at rate at which the learning algorithm cannot keep up. At around 200 trials, the algorithm has learned all available actions well enough to start to exploit, and there is an attendant rise in performance; however, each time a new action arrives, both exploitation and performance drop temporarily as new learning must ensue.

At the $T = 5000$ timescale (middle panel, 14 actions added), exploitation rates are consistently higher (approaching 0.6 or 60% of the trials), and performance is beginning to converge to the optimal. New action arrivals still cause temporary dips, but overall upward

progress is setting in.

At $T = 25,000$ (right panel, 27 actions added), the algorithm is exploiting over 80% of the time, and performance has converged to optimal up to the $\epsilon = 0.1$ accuracy set for the KWIK subroutine. If ϵ tends to 0 as T increases, as in the formal analysis, we eventually converge to 0 regret.

3.6. Detailed Proofs

3.6.1. Proof of Theorem 3.4

We now show that relaxing sublinearity of $N(t)$ is insufficient to imply no-regret on MAB.

Restatement of Theorem 3.4:
There exists a class $\mathcal{F}$ that is KWIK-learnable with a don't-know bound of 1 such that if $N(t) = t$, for any learning algorithm A and any T, there is a sequence of trials in the arriving action model such that $R_A(T)/T > c$ for some constant $c > 0$.

Proof. Let $\mathcal{A} = \mathbb{N}$ and $e : \mathbb{N} \to \mathbb{R}^+$ be a fixed encoding function satisfying $e(n) \leq \gamma$ for any n, and let d be a corresponding decoding function satisfying $(d \circ e)(n) = n$.

Consider $\mathcal{F} = \{f_n \mid n \in \mathbb{N}\}$, where $f_n(n) = 1$ and $f_n(n') = e(n)$ for all other n'. The class $\mathbb{N}$ is KWIK-learnable with at most a single $\perp$ in the noise-free case. Observing $f_n(n')$ for an unknown f_n and arbitrary $n' \in \mathbb{N}$ immediately reveals the identity of f_n. Either $f_n(n') = 1$, in which case $n = n'$, or else $n = d(f_n(n'))$.

Let $\mathcal{A}$ and $\mathcal{F}$ be as just described. There exists an absolute constant $c > 0$ such that for any $T \geq 4$, there exists a sequence $\{n_t, S^t\}$ satisfying $N(T) = T$, and $R_A(T)/T > c$ for any A.

Let σ be a random permutation of $\{1, ..., T\}$, and S^1 be the ordered set $\{f_{\sigma(1)}, f_{\sigma(2)}, ..., f_{\sigma(T)}\}$. In other words, the actions $f_1, ..., f_T$ are shuffled, and immediately presented to the algorithm on the first round. $S^t = \emptyset$ for $t > 1$. Let n_t be drawn uniformly at random from

$\{1, ..., T\}$ on each round t.

Immediately, we have that $E\left[\sum_{t=1}^{T} \max_{f \in \mathcal{F}^t} f(n_t)\right] = T$ since $F^t = \{1, ..., T\}$ for all t.

Now consider the actions $\{\hat{f}_t\}$ selected by an arbitrary algorithm A. Define $\hat{F}(\tau) = \{\hat{f}_t \in \mathcal{F} \mid t < \tau\}$, the actions that have been selected by A before time τ. Let $U(\tau) = \{n \in \mathbb{N} \mid f_n \in \hat{F}(\tau)\}$ be the states n, such the corresponding best action f_n has been used in the past, before round τ. Also let $\bar{\mathcal{F}}(\tau) = \{1, ..., T\} \setminus \hat{\mathcal{F}}(\tau)$.

Let R_τ be the reward earned by the algorithm at time τ. If $n_\tau \in U(\tau)$, then the algorithm has played action f_{n_τ} in the past, and knows its identity. Therefore, it may achieve $R_\tau = 1$. Since n_τ is drawn uniformly at random from $\{1, ..., T\}$, $P(n_\tau \in U(\tau) \mid U(\tau)) = \frac{|U(\tau)|}{T}$. Otherwise, in order to achieve $R_\tau = 1$, any algorithm must select $\hat{f}_\tau$ from amongst $\bar{F}(\tau)$. But since the actions are presented as a random permutation, and no action in $\bar{F}(\tau)$ has been selected on a previous round, any such assignment satisfies $P(\hat{f}_\tau = f_{n_\tau} \mid n_\tau \notin U(\tau)) = \frac{1}{T - |U(\tau)|}$.

Therefore for any algorithm we have:

$$\begin{aligned}
&P(R_\tau = 1 \mid U(\tau)) \\
&\leq P(n_\tau \in U(\tau) \mid U(\tau)) + P(n_\tau \notin U(\tau), \hat{f}_\tau = f_{n_\tau} \mid U(\tau)) \\
&\leq \frac{|U(\tau)|}{T} + \left(1 - \frac{|U(\tau)|}{T}\right)\left(\frac{1}{T - |U(\tau)|}\right)
\end{aligned}$$

Note that the RHS of the last expression is a convex combination of 1 and $\left(\frac{1}{T-|U(\tau)|}\right) \leq 1$, and is therefore increasing as $\frac{|U(\tau)|}{T}$ increases. Since $|U(\tau)| < \tau$ with probability 1, we have:

$$P(R_\tau = 1) \leq \frac{\tau}{T} + \left(1 - \frac{\tau}{T}\right)\left(\frac{1}{T - \tau}\right) \tag{3.3}$$

Let $Z(T) = \sum_{\tau=1}^{T} I(R_\tau = 1)$, count the number of rounds on which $R_\tau = 1$. This gives us:

$$\begin{aligned} E[Z(T)] &= \sum_{\tau=1}^{T} P(R_\tau = 1) \\ &\leq \frac{T}{2} + \sum_{\tau=1}^{T/2} P(R_\tau = 1) \\ &\leq \frac{T}{2} + \frac{T}{2}\left[\frac{T}{2T} + \left(1 - \frac{T}{2T}\right)\left(\frac{1}{T - T/2}\right)\right] \end{aligned}$$

Where the last inequality follows from the fact that equation 3.3 is increasing in τ.

Thus $E[Z(T)] \leq \frac{3T}{4} + \frac{1}{2}$. On rounds where $R_\tau \neq 1$, R_τ is at most γ, giving:

$$R_A(T)/T \geq 1 - \left[\frac{3}{4} + \frac{1}{2T} + \frac{\gamma}{4}\right]$$

Taking $T \geq 4$, gives us:

$R_A(T)/T \geq \frac{1}{8} - \frac{\gamma}{4}$. Since γ is arbitrary we have the desired result.

□

3.6.2. Proof of Theorem 3.5

We now show that relaxing KWIK to supervised no-regret is insufficient to imply no-regret on MAB.

Restatement of Theorem 3.5:
There exists a class $\mathcal{F}$ that is supervised no-regret learnable such that if $N(t) = \sqrt{t}$, for any learning algorithm A and any T, there is a sequence of trials in the arriving action model such that $R_A(T)/T > c$ for some constant $c > 0$.

Proof. First we describe the class $\mathcal{F}$. For any n-bit string x, let f_x be a function such that $f_x(x)$ is some large value, and for any $x' \neq x$, $f_x(x') = 0$. It's easy to see that $\mathcal{F}$ is

not KWIK learnable with a polynomial number of don't-knows — we can keep feeding an algorithm different inputs $x' \neq x$, and as soon as the algorithm makes a prediction, we can reselect the target function to force a mistake (this is essentially the same argument proving that monomials are not KWIK learnable). $\mathcal{F}$ is no-regret learnable, however: we just keep predicting 0. As soon as we make a mistake, we learn x, and we'll never err again, so our regret is at most $O(1/T)$.

Now in the arriving action model, suppose we initially start with r distinct functions/actions $f_i = f_{x_i} \in \mathcal{F}$, $i = 1, \ldots, r$. We will choose $N(T) = \sqrt{T}$, which is sublinear, and $r = \sqrt{T}$, and we can make T as large as we want. So we have a no-regret-learnable $\mathcal{F}$ and a sublinear arrival rate; now we argue that the arriving action MAB problem is hard.

Pick a random permutation of the f_i, and let i be the indices in that order for convenience. We start the task sequence with all x_1's. The MAB learner faces the problem of figuring out which of the unknown f_is has x_1 as its high-payoff input. Since the permutation was random, the expected number of assignments of x_1 to different f_i before this is learned is $r/2$. At that point, all the learner has learned is the identify of f_1 — the fact that it learned that other $f_i(x_1) = 0$ is subsumed by learning $f_1(x_1)$ is large, since the f_i are all distinct.

We then continue the sequence with x_2's until the MAB learner identifies f_2, which now takes $(r-1)/2$ assignments in expectation. Continuing in this vein, the number of assignments made before learning (say) half of the f_i is $\sum_{j=1}^{r/2}(r-j)/2 \approx \Omega(r^2) \approx \Omega(T)$. On this sequence of $\Omega(T)$ tasks, the MAB learner will have gotten non-zero payoff on only $r = \sqrt{T}$. The offline optimal, on the other hand, always knows the identity of the f_i and gets large payoff on every single task. So any learner's cumulative regret to offline grows linearly with T. $\square$

CHAPTER 4 : Large Scale Bandits : Lower Bounds on Regret

In this chapter we consider the MAB problem for a *general* function class $\mathcal{F}$ (in contrast to previous chapters where $\mathcal{F}$ admitted a graphical decomposition, or was KWIK-learnable), and focus on the number of rounds required to achieve low regret. However, we will fix our attention on the non-contextual version of the problem. We give a characterization in terms of a new measure of the complexity of the function class $\mathcal{F}$ that we call the *haystack dimension*, which intuitively captures the extent to which maximizing a function via queries requires a search for a small number of items (needles) amongst a much larger number of otherwise undifferentiated possibilities (a haystack). We then give upper and lower bounds involving the haystack dimension of $\mathcal{F}$ that are within a $\log|\mathcal{F}|$ factor. Note that for the hardest MAB problems — where the haystack dimension can be as large as $|\mathcal{F}|$ — this logarithmic factor is relatively benign. Our main results are graphically summarized in Figure 2.

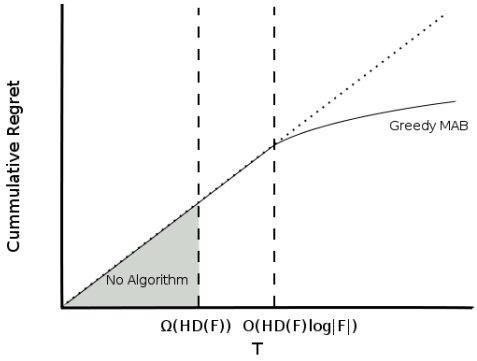

Figure 2: A graphical summary of the main results of this chapter. This figure illustrates that *any* MAB algorithm for a function class $\mathcal{F}$ must suffer linear regret for a number of rounds on the order of the haystack dimension (denoted $\mathrm{HD}(\mathcal{F})$), while the MAB algorithm presented in this chapter, called Greedy MAB, begins to suffer sublinear regret after roughly $\mathrm{HD}(\mathcal{F})\log|\mathcal{F}|$ rounds.

An interesting aspect of our methods is the connection drawn between MAB problems and the problem of exact learning of functions from queries. We observe that functional MAB problems implicitly embed the problem of finding a maximum of an unknown function in $\mathcal{F}$ from only input-output queries (generalizations of membership queries), which may or

may not be much easier than exact learning. Our analysis shows that any functional MAB algorithm must implicitly be willing to trade off between two distinct types of queries: *max queries* (which attempt to directly guess the maximum of f^*) and *information queries* (which attempt to make progress by reducing the version space, as is traditional in many query learning models). We show that either one of these query types (and essentially no others) can result in progress towards finding the maximum. The haystack dimension can then be viewed as a measure of the extent to which progress can be made at any step via either one of these query types.

Our characterization holds for any finite-cardinality $\mathcal{F}$ (though even the number of *actions* may still be infinite), but we also describe a generalization to the case of infinite $\mathcal{F}$ via covering techniques (which in general does not provide as tight bounds as its finite-cardinality specialization). We also stress that our results only apply to query complexity and regret; we make no claims about computational efficiency (necessarily, due to the generality of our setting). In this sense, the haystack dimension can be seen as playing a role in the study of functional MAB problems analogous to that played by quantities such as VC dimension and teaching dimension in other learning models, which also characterize sample or informational complexity, but not computational complexity.

4.1. Related Work

Many authors (at least since Thompson (1933)) have studied finite MAB problems where the action payoffs are assumed to be correlated; see Mersereau et al. (2009, p. 4) for an excellent survey. As previously discussed, more recent work has focused on infinite MAB problems where the action payoffs are related via an unknown function belonging to a known function class, such as the set of all Lipschitz continuous functions (Kleinberg et al., 2008; Bubeck et al., 2008; Slivkins, 2014). Compared to previous work, our results provide a complete analysis for significantly more general function classes.

Obviously, maximizing an unknown function via queries is no harder than exactly learning

the function. Hegedüs (1995) characterized the query complexity of exact learning in terms of the *extended teaching dimension* of the function class. For some restricted function classes the haystack dimension and extended teaching dimension coincide[1], and in these cases our analysis approximately recovers the bounds due to Hegedüs (1995), but with a significant advantage: our lower bound holds for all *randomized* algorithms, while the earlier bound only applied to deterministic algorithms.

A variant of exact learning (but not maximization) of functions has been considered under the name of *generalized binary search*. Nowak (2009) provided an analysis that only applies under a certain technical condition. In the language of this work, the condition implies that the haystack dimension is a constant independent of the structure of the function class. In contrast, our analysis applies to any $\mathcal{F}$ and considers the maximization problem and its relationship to MAB directly.

4.2. Functional Bandits (MAB) and Maximizing From Queries (MAX)

A *functional MAB problem* is defined by a set of *actions* $\mathcal{X}$ and a set of possible *payoff functions* $\mathcal{F}$. A target payoff function $f^* \in \mathcal{F}$ is selected. In each round $t = 1, 2, \ldots$, an algorithm selects an action x_t, and receives an independent payoff from a distribution whose support is contained in a bounded interval, and has mean $f^*(x_t)$. The goal of the algorithm is to receive nearly as much cumulative payoff as an algorithm that selects the best action every round. More precisely, the worst-case expected *regret* of algorithm A in round T is $R_A(T) \triangleq \sup_{f^* \in \mathcal{F}} E\left[T \cdot \sup_{x \in \mathcal{X}} f^*(x) - \sum_{t=1}^{T} f^*(x_t)\right]$, where the expectation is with respect to the random payoffs and any internal randomization of the algorithm. We say that algorithm A is *no-regret* if $\lim_{T\to\infty} R_A(T)/T = 0$.

For any functional MAB problem, we can define a corresponding and (as we shall see) closely related functional *maximizing from queries* (or MAX) problem: In each round $t = 1, 2, \ldots$ an algorithm A selects a *query* $x_t \in \mathcal{X}$, and then observes $y_t = f^*(x_t)$. Letting X^f be the

[1] Essentially just those classes for which maximizing an unknown function is as difficult as exactly learning it; see Example 4.2.

set of maxima of the function f (assumed to be non-empty), the goal of the algorithm is to eventually select a $x \in X^{f^*}$. Let $T^{A,f^*} = \min\{t : x_t \in X^{f^*}\}$ be the first round such that x_t is a maximum of f^*. We are interested in bounding the worst-case expected *query complexity* $Q_A \triangleq \sup_{f^* \in \mathcal{F}} E[T^{A,f^*}]$, where the expectation is with respect to any internal randomization of the algorithm.

It is important to note that, in a functional MAB problem, in each round t the algorithm only observes a *sample* from a distribution with mean $f^*(x_t)$, while in a functional MAX problem, the algorithm observes $f^*(x_t)$ *directly*. In Sections 4.3–4.6, we characterize the query complexity of the MAX problem for $\mathcal{F}$, and then apply these results in Sections 4.8–4.9 to characterize the optimal regret for the corresponding MAB problem for $\mathcal{F}$. Consequently, the analysis in Sections 4.3–4.6 will not deal with stochasticity, which is addressed afterwards. Also, we refer to elements of $\mathcal{X}$ as *actions* in the MAB context, but as *queries* in the MAX context — this difference exists only to agree with historical usage.

4.3. The Haystack Dimension

In this section we give the definition of the haystack dimension for function classes $\mathcal{F}$ of *finite cardinality*; generalization to the infinite case is given later.

The formal definition of the haystack definition requires some notation and machinery, but the intuition behind it is rather simple, so we first describe it informally. In words, the haystack dimension identifies the "worst" subset of $\mathcal{F}$, in the sense that on that subset, no matter what query is made and no matter what response is received, only a small fraction θ of the functions in the subset are eliminated due to inconsistency with the query, or are maximized by the query. It turns out mathematically that the right definition of the haystack dimension is the inverse quantity $1/\theta$ for this worst subset. We now proceed with the formal definition.

In the context of a MAX problem, a query $x \in \mathcal{X}$ can be thought of as providing information about the identity of f^*. In particular, f^* cannot be any of the functions in $\mathcal{F}$ inconsistent

with with the value $f^*(x)$ observed at x. So one strategy for finding an element of X^{f^*} is to first issue a sequence of *information queries*, that uniquely identify f^*, and then select any $x \in X^{f^*}$.[2]

However, sometimes identifying the true function f^* exactly requires many more queries than necessary for maximization. For example, if many functions $f \in \mathcal{F}$ are maximized by one particular query, it may be useful to play such a *max query*, even if it is not particularly useful for learning f^*.

In the extreme case, there might exist an $x^* \in \mathcal{X}$ such that $x^* \in X^f$ for *all* $f \in \mathcal{F}$. In this case, an element of X^{f^*} can be selected in one query, without ever needing to identify f^*. On the other hand, there are also $\mathcal{F}$ for which exact learning is the fastest route to maximization. [3]

Any general algorithm for maximization from queries thus needs to be implicitly able to consider queries that would eliminate many candidate functions, as well as queries that might be an actual maximum.

Before continuing, we define some convenient notation that we will use throughout the rest of the chapter. For any set $F \subseteq \mathcal{F}$, define the *inconsistent set* $F(\langle x, y \rangle) \triangleq \{f \in F : f(x) \neq y\}$ to be the functions in F that are inconsistent with the query-value pair $\langle x, y \rangle$. Also, for any set $F \subseteq \mathcal{F}$, define the *maximum set* $F(x) \triangleq \{f \in F : x \in X^f\}$ to be the functions in F for which x is a maximum.

Intuitively, the *haystack dimension* $\mathrm{HD}(\mathcal{F})$ of a function class $\mathcal{F}$ will characterize a subset of $\mathcal{F}$ on which no query is effective in the two senses previously discussed. For a subset $F \subseteq \mathcal{F}$, let:

$$\rho(F, x) = \inf_y \frac{|F(x) \cup F(\langle x, y \rangle)|}{|F|} \quad \text{and} \quad \rho(F) = \sup_{x \in \mathcal{X}} \rho(F, x).$$

$\rho(F, x)$ is the fraction of functions in F, which are guaranteed to be maximized, or deemed

[2] In the case of boolean functions or concepts, identifying f^* exactly is the problem of *learning from membership queries* (Angluin, 1988).

[3] See Example 4.2.

inconsistent, by the query x, for the worst-case possibility for y. If $\rho(F)$ is small, no query is guaranteed to be effective as either a max or information query on the subset F. Now let $F_\theta = \arg\inf_{F \subseteq \mathcal{F}} \rho(F)$ and $\theta = \rho(F_\theta)$.

Definition 4.1. *Let F_θ and θ be defined as above. Then the haystack dimension* $\mathrm{HD}(\mathcal{F})$ *of $\mathcal{F}$ is defined as $\frac{1}{\theta}$.*

Note that the haystack dimension can be as small as 1 (all functions share a common maximum-output input) and as large as $|\mathcal{F}|$ (every query eliminates at most one function in the senses discussed, the canonical "needle in a haystack").

While we will not describe the construction in detail, we note that the Haystack Dimension can be generalized to any sequential problem where the goal of the learner is to query an input x satisfying some property (in the above, the property is $x \in X^f$). Our main theorems for the MAX setting (4.1 and 4.2) generalize immediately when considering some other property. Furthermore, we need not restrict our observation model to input-output queries (where the learner observes only $f^*(x)$). In particular, $F(\langle x, y \rangle)$ can be defined as inconsistency with respect to whatever information the learner receives about f^*.

4.4. Examples of the Haystack Dimension

In this section, we provide a few function classes which help illustrate how the haystack dimension characterizes the difficulty of maximizing an unknown $f^* \in \mathcal{F}$. Many of these examples will be useful for subsequent constructions in the chapter.

The first construction considered is the "needle in a haystack". Fix a finite $\mathcal{X}$. For each $x \in \mathcal{X}$, let f_x be the function defined to have $f_x(x) = 1$ and $f_x(x') = 0$ for all $x' \neq x$. Now let $\mathcal{H}_\mathcal{X} = \{f_x \mid x \in \mathcal{X}\}$.

Example 4.1. $HD(\mathcal{H}_\mathcal{X}) = |\mathcal{H}_\mathcal{X}|$

Proof. For any $x \in \mathcal{X}$, f_x is the only function in $\mathcal{H}_\mathcal{X}$ that attains its max at x. Furthermore, all other functions output a 0 on input x. Therefore, letting $F = \mathcal{H}_\mathcal{X}$, $F(x) = \{f_x\}$ and

$F(\langle x, 0\rangle) = \{f_x\}$ for any $x \in \mathcal{X}$. This implies that $\rho(\mathcal{H}_\mathcal{X}) = |\mathcal{H}_\mathcal{X}|^{-1}$. Since $\rho(F)$ cannot be smaller than this quantity for any $F \subseteq \mathcal{F}$, the haystack dimension of $\mathcal{H}_\mathcal{X}$ indeed equals $|\mathcal{H}_\mathcal{X}|$. □

When $f^* \in \mathcal{H}_\mathcal{X}$, maximizing and learning f^* coincide, and both amount to guessing the x for which $f^*(x) = 1$. We now describe a function class $\mathcal{G}$ in which any algorithm is essentially forced to learn the true function f^*.

The input space $\mathcal{X}$ will consist of two components — $\mathcal{X}_1$ and $\mathcal{X}_2$, with $\mathcal{X}$ being the union of these disjoint domains. The high-level idea is to "marry" a small shattered set (in the sense of VC dimension) to a much larger haystack construction. Subdomain $\mathcal{X}_1$ consists of n points $\{a_0, ..., a_{n-1}\}$. We construct $\mathcal{G}$ as follows. On $\mathcal{X}_1$, all possible binary labelings of the n points appear in $\mathcal{G}$, giving a total of 2^n functions. Let us think of each function in $\mathcal{G}$ as being equated with the integer given by its binary labeling of the points in $\mathcal{X}_1$. So a function f is equated with the integer $z(f) = \sum_{i=0}^{n-1} 2^i f(a_0)$. Now let the much larger set $\mathcal{X}_2 = \{0, \ldots, 2^n - 1\}$, and for any $x \in \mathcal{X}_2$ define $f(x) = 2$ if $x = z(f)$ and $f(x) = 0$ otherwise.

Thus, the behavior of a function on $\mathcal{X}_1$ entirely defines the function on $\mathcal{X}_2$ as well, and the labeling on $\mathcal{X}_1$ gives us the index of the function's maximum, which is always equal to 2 and occurs at exactly one point in $\mathcal{X}_2$ (determined by the index).

Note that there is an algorithm that finds the max of f^* in $O(n)$ queries simply by querying every input in $\mathcal{X}_1$ and learning the identity of f^* exactly. Intuitively, no algorithm can do much better. To see why, suppose f^* were drawn uniformly at random from $\mathcal{G}$. Note that an action $r \in \mathcal{X}_2$ has an exponentially small probability of being the function's max and, in the event that $f^*(r) = 0$, only serves to inform the algorithm that f^* is not the single $f \in \mathcal{G}$ with $z(f) = r$. Also, observe that if the "zooming" algorithm of Kleinberg et al. (2008) is applied to $\mathcal{G}$, it will take exponential time in the worst-case to find a maximum of f^*, essentially because it makes no attempt to exploit the special structure of $\mathcal{G}$.

Example 4.2. $HD(\mathcal{G}) = \Theta(n) = \Theta(\log |\mathcal{G}|)$

Proof. We first show that there is an $F \subseteq \mathcal{G}$ with $\rho(F) = \frac{1}{n}$. For each $x \in \mathcal{X}_1$, let f_x be the function which outputs $f(x) = 1$, and $f(x') = 0$ for all $x' \in \mathcal{X}_1$, $x \neq x'$. Let $F = \{f_x \in \mathcal{G} \mid x \in \mathcal{X}_1\} = \{f \mid z(f) \in \{2^0, 2^1, ..., 2^{n-1}\}\}$. $|F| = n$.

Consider any query $x \in \mathcal{X}_1$. $F(x) = \emptyset$, since no functions achieve their maximum on a query in $\mathcal{X}_1$. Furthermore, $F(\langle x, 0\rangle) = \{f_x\}$, since f_x is the only function in F which doesn't output a 0 on query x. Thus $\rho(F, x) = \frac{1}{n}$ for any $x \in \mathcal{X}_1$. For a query $r \in \mathcal{X}_2$, $\rho(F, r) \leq \frac{1}{n}$, since at most one function in F (and $\mathcal{G}$) achieves its maximum at r, and all other functions output a zero at r. Thus, $\rho(F) = \frac{1}{n}$.

We now argue that for an arbitrary $F \subseteq \mathcal{G}$, $\rho(F) \geq \frac{1}{2n}$. Let $r_1(x) = |f \in F : f(x) = 1|/|F|$, be the fraction of functions that exhibit a 1 at action $x \in \mathcal{X}_1$. Let $r_0(x) = 1 - r_1(x)$. Suppose there is an action $x \in \mathcal{X}_1$ such that $r_1(x) > \frac{1}{2n}$ and $r_0(x) > \frac{1}{2n}$. Then, at least $\frac{1}{2n}$ of the functions in F would be inconsistent with any observed output. That is, both $\frac{|F(\langle x,0\rangle)|}{|F|} > \frac{1}{2n}$ and $\frac{|F(\langle x,1\rangle)|}{|F|} > \frac{1}{2n}$.

Otherwise, for every x in $\mathcal{X}_1$, either $r_1(x) \leq \frac{1}{2n}$ or $r_0(x) \leq \frac{1}{2n}$ (i.e. one outcome is quite rare). This implies that at least $1/2$ of the functions in F exhibit the same behavior on all x in $\mathcal{X}_1$. However, unless F is a singleton set, this cannot occur since each $f \in \mathcal{G}$ exhibits unique behavior on $\mathcal{X}_1$. □

The preceding example illustrates a function class for which any algorithm querying for the max must ultimately learn the true function f^*. However, the opposite extreme is also possible. Consider a function class $\mathcal{G}_{\max}$. Let there be a distinguished $x^* \in \mathcal{X}$ such that every $f \in \mathcal{G}_{\max}$ attains its maximum at x^*. It may be arbitrarily difficult to learn the behavior of f^* on the remainder of $\mathcal{X}$. However, finding the maximum can be done trivially in a single query.

Example 4.3. $HD(\mathcal{G}_{\max}) = 1$

Proof. For every $F \subseteq \mathcal{G}_{\max}$ $F(x^*) = F$, and the proof is immediate. □

Finally, there are function classes which require a hybrid between learning and maximization. We construct such a class, $\mathcal{G}^+$. Let $\mathcal{X}$ be the disjoint union of three sets $\mathcal{X}_1 \cup \mathcal{X}_2 \cup \mathcal{X}_3$. We let $|\mathcal{X}_1| = n$, $|\mathcal{X}_2| = m$ and create a function in $\mathcal{G}^+$ for each binary labeling of $\mathcal{X}_1$ and $\mathcal{X}_2$, as in the construction of $\mathcal{G}$. We let $z_1(f)$ be the integer corresponding to the labeling f gives X_1 and $z_2(f)$ be the integer corresponding to the labeling f gives X_2.

Now let $\mathcal{X}_3 = M_0 \cup M_1 \cup ... \cup M_{2^n-1}$ where each $|M_i| = c$. Again, like in the construction of $\mathcal{G}$, for each $f \in \mathcal{G}^+$, there will be a single $r \in \mathcal{X}_3$ such that $f(r) = 2$ and $f(r) = 0$ otherwise. However, this time, the maximizing element of f will be found in $M_{z_1(f)}$. And the particular element of $M_{z_1(f)}$ will be given by $z_2(f) \mod c$.

If we think of $c < n < m$, then there is an $n + c$ algorithm for finding the max of $\mathcal{F}$. The algorithm learns the set M_i which contains the max of f^* by querying each element of $\mathcal{X}_1$. It then tries each of the c elements in M_i. Note that the true identity of f^* is never learned, and the "interesting" learning problem is discovering the set M_i containing the maximizing action (i.e., learning the behavior on $\mathcal{X}_1$).

Example 4.4. *If $c < n < m$, $HD(\mathcal{G}^+) = \Theta(n) = \Theta(\log |\mathcal{G}^+|)$*

Proof. We sketch the proof which proceeds almost identically to that of Example 4.2. There is an F with that $\rho(F) = \frac{1}{n}$. F is identical to that used in Example 4.2 on $\mathcal{X}_1$. The behavior on $\mathcal{X}_2$ is identical across all functions, and the behavior on $\mathcal{X}_3$ is determined by these choices.

To see that $\rho(F) \geq \frac{1}{2n}$, we use the same reasoning as Example 4.2. However, if for every x in $\mathcal{X}_1$, either $r_1(x) \leq \frac{1}{2n}$ or $r_0(x) \leq \frac{1}{2n}$, rather than implying a contradiction, this implies that there is actually a query $x^* \in \mathcal{X}_3$ such that x^* maximizes at least a $\frac{1}{2c}$ fraction of the functions in F. □

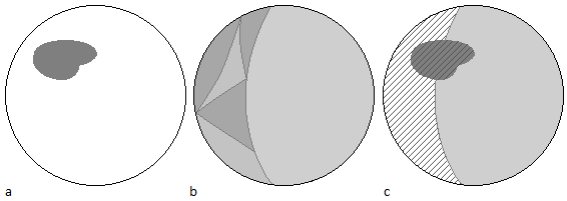

Figure 3: (a) The shaded region represents the subset of functions which attain a maximum at some input x. (b) Querying x also induces a partition of the function space, in which each piece of the partition contains functions that output the same value on x. The least informative query response is thus the largest piece of this partition. (c) Upon querying x, the greedy algorithm eliminates from its attention space a set *at least* the size of the striped region. This region is the union of the maximizing set and the *complement* of the largest partition piece.

4.5. MAX Query Complexity Upper Bound

Before arguing that the haystack dimension gives lower bounds on the query complexity of any algorithm on $\mathcal{F}$, we observe that it motivates a simple, natural, greedy algorithm. In each round t, the *greedy algorithm* G selects $x_t = \arg\sup_{x \in \mathcal{X}} \rho(F_t, x)$ where the *attention space* F_t is defined inductively as follows: $F_1 = \mathcal{F}$ and $F_{t+1} = F_t \setminus (F_t(x_t) \cup F_t(\langle x_t, y_t \rangle))$. (Note that for fixed F_t, $\rho(F_t, x)$ takes values in $\{1/|F_t|, 2/|F_t|, ..., 1\}$, so the supremum is achieved by an $x \in \mathcal{X}$.)

Essentially, the greedy algorithm always selects the query that results in the largest guaranteed reduction of the attention space [4]; see Figure 3. It is important to note that the attention space differs from the version space of traditional learning in that it *excludes functions that may still be consistent with the observed data*, but for which the algorithm has already played a query which would have maximized such functions. Indeed, in the extreme case the algorithm may choose to query x such that *all functions remaining* have the same output on x — in which case the query conveys no "information" in the traditional learning sense, but nevertheless functions attaining their maximum at x will be discarded.

Theorem 4.1. *Let G be the greedy algorithm for the MAX problem for $\mathcal{F}$. Then $Q_G \leq \mathrm{HD}(\mathcal{F}) \log |\mathcal{F}|$.*

[4] Note that we assume the specified computations can in fact be implemented in finite computation time.

Proof. We know that the attention space F_t is nonempty for all rounds $t \leq Q_G$ (otherwise the greedy algorithm would have selected a maximum of f^* before round Q_G). And at least a θ fraction of the attention space is removed in each round t, because $\theta = \rho(F_\theta) \leq \rho(F_t) = \rho(F_t, x_t) \leq \frac{|F_t(x_t) \cup F_t(\langle x_t, y_t \rangle)|}{|F_t|}$ by the definition of θ, F_θ, and x_t. Thus we have $(1-\theta)^{Q_G}|\mathcal{F}| \geq 1$. Taking the logarithm of both sides of this inequality, applying $\log(1-x) \leq -x$ for $x < 1$, rearranging, and noting that $\mathrm{HD}(\mathcal{F}) \triangleq \frac{1}{\theta}$ by definition implies the theorem. □

4.6. MAX Query Complexity Lower Bound

In this section we prove that that haystack dimension in fact provides a lower bound on the query complexity of *any* algorithm finding a maximum of $f^* \in \mathcal{F}$:

Theorem 4.2. *Let A be any algorithm for the MAX problem for $\mathcal{F}$. Then $Q_A = \Omega(\mathrm{HD}(\mathcal{F}))$.*

The proof of Theorem 4.2 is somewhat involved and developed in a series of lemmas, so we shall first sketch the broad intuition. The lower bound will draw the target f^* uniformly at random from F_θ, where F_θ is as in the definition of haystack dimension, and is the subset of functions on which the greedy algorithm guarantees the least amount of progress (in terms of reducing its attention space, as defined above). By definition no other algorithm A can make more progress than θ on its first query starting from F_θ (since this is the maximum possible, and obtained by greedy). After the first step, the greedy algorithm and algorithm A may have different attention spaces, and thus on subsequent steps A may make greater progress than greedy; but A cannot make "too much" progress on (say) its second step, since otherwise its query there would have made more than θ progress on F_θ. This insight leads to a formal recurrence inequality governing the progress rate of A, whose solution, when combined with a Bayesian argument, leads to a lower bound that is $\Omega(\frac{1}{\theta}) \triangleq \Omega(\mathrm{HD}(\mathcal{F}))$. We now proceed with the formal development.

Suppose we have an algorithm A that, given access to query-value pairs from f^*, generates a sequence of queries $\{x_t\}$ (possibly depending on its random bits). Let $y_t = f^*(x_t)$. Let $H_1 \triangleq F_\theta$ and

$$y_t^-(x) \triangleq \arg\inf_y |H_t(x) \cup H_t(\langle x, y\rangle)| \qquad S_t(x) \triangleq H_t(x) \cup H_t(\langle x, y_t^-(x)\rangle)$$
$$H_{t+1} \triangleq H_t \setminus S_t(x_t) \qquad x_t^* \triangleq \arg\sup_{x\in\mathcal{X}} |S_t(x)| \qquad \delta_t \triangleq \frac{|S_t(x_t^*)|}{|H_t|}$$

These definitions are closely related to those for the greedy algorithm and the haystack dimension. $y_t^-(x)$ is the "least helpful" possible output value on query x, H_t is the attention space of algorithm A when starting from F_θ if only least helpful outputs are returned, $S_t(x)$ is the set of functions in H_t that either attain a maxima at x or fall outside the largest partition induced by x, and δ_t is the progress (fractional reduction of the attention space) made by the greedy algorithm.

Our first lemmas, which codify the aforementioned Bayesian argument, show that when f^* is drawn uniformly from F_θ, the probability a deterministic algorithm (a restriction removed shortly) finds a maximum in fewer than T steps is bounded by the sum of the δ_t.

Lemma 4.1. *Fix a sequence $\{x_t\}$. Let $B_t = \{f \in F_\theta : x_s \notin X^f \wedge y_s^-(x_s) = f(x_s) \forall s < t\}$. Then $H_t = B_t$ for all t.*

Proof. When $t = 1$, $B_1 = \{f \in F_\theta\}$ and the claim is immediate. Otherwise, assume that $H_t = B_t$ for some $t \geq 1$. We have:

$$\begin{aligned} B_{t+1} &= B_t \cap \{f \in F_\theta : x_t \notin X^f \wedge y_t^-(x_t) = f(x_t)\} \\ &= B_t \setminus \{f \in B_t : x_t \in X^f \vee y_t^-(x_t) \neq f(x_t)\} = H_t \setminus S_t(x_t) = H_{t+1} \end{aligned}$$ □

Lemma 4.2. $\Pr_{f^*\sim U_{F_\theta}}\left[T^{A,f^*} < T\right] \leq \sum_{t=1}^{T} \delta_t$ *for any deterministic algorithm A and constant T.*

Proof. Since A is deterministic, the sequence $\{x_t\}$ is determined by the choice of $f^* \in \mathcal{F}$. By Lemma 4.1 we have $H_t = \{f \in F_\theta : x_s \notin X^f \wedge y_s^-(x_s) = f(x_s)\ \forall s < t\}$ for all t. Moreover, since $y_s^-(x_s) = f^*(x_s)$ for all $s < t$ and $f^* \in \mathcal{F}$ and algorithm A is deterministic, the sequence $\{x_s\}_{s\leq t}$ is identical for any choice of $f^* \in H_t$. Let U_F denote the uniform

distribution over F. We have:

$$\begin{aligned}
\Pr_{f^* \sim U_{F_\theta}}\left[T^{A,f^*} < T\right] &= \Pr_{f^* \sim U_{F_\theta}}\left[\exists t < T : x_t \in X^{f^*}\right] \leq \Pr_{f^* \sim U_{F_\theta}}\left[\exists t < T : x_t \in X^{f^*} \vee y_t^-(x_t) \neq f^*(x_t)\right] \\
&\leq \sum_{t=1}^{T} \Pr_{f^* \sim U_{F_\theta}}\left[x_t \in X^{f^*} \vee y_t^-(x_t) \neq f^*(x_t) \mid x_s \notin X^{f^*} \wedge y_s^-(x_s) = f^*(x_s) \forall s < t\right] \\
&= \sum_{t=1}^{T} \Pr_{f^* \sim U_{H_t}}\left[x_t \in X^{f^*} \vee y_t^-(x_t) \neq f^*(x_t)\right] \leq \sum_{t=1}^{T} \frac{|S_t(x_t)|}{|H_t|} \leq \sum_{t=1}^{T} \delta_t
\end{aligned}$$

□

We thus see that one approach to lower bounding T^{A,f^*} is to bound the growth rate of the sequence $\{\delta_t\}$. Intuitively, we should expect that $(1-\delta_t)\delta_{t+1} \leq \delta_t$. To see why, recall that δ_t is the most progress that can be guaranteed by a single query if the function is drawn from the space H_t. This is the query that would be selected by the greedy algorithm if it were run on H_t. Suppose that it were instead that case that $(1-\delta_t)\delta_{t+1} > \delta_t$. By playing x^*_{t+1} on H_t at least $(1-\delta_t)\delta_{t+1}$ fraction of the functions in H_t would either attain a maximum point at x^*_{t+1} or be eliminated as inconsistent with the observed value $f^*(x^*_{t+1})$. Thus the query x^*_t, which only guaranteed that a δ_t fraction of the functions in H_t have this property, was suboptimal, a contradiction . More formally:

Lemma 4.3. *$(1-\delta_t)\delta_{t+1} \leq \delta_t$ for all t.*

Proof.

$$\begin{aligned}
(1-\delta_t)\delta_{t+1} &= \left(1 - \frac{|S_t(x^*_t)|}{|H_t|}\right)\frac{|S_{t+1}(x^*_{t+1})|}{|H_{t+1}|} = \left(\frac{|H_t| - |S_t(x^*_t)|}{|H_t|}\right)\frac{|S_{t+1}(x^*_{t+1})|}{|H_t| - |S_t(x_t)|} \\
&\leq \frac{|S_{t+1}(x^*_{t+1})|}{|H_t|} \leq \frac{|S_t(x^*_{t+1})|}{|H_t|} \leq \frac{|S_t(x^*_t)|}{|H_t|} = \delta_t
\end{aligned}$$

Here the first inequality follows from the the definition of x^*_t as a maximizer of $|S_t(x)|$ (and thus $(|H_t| - |S_t(x^*_t)|)/(|H_t| - |S_t(x_t)|) \leq 1$). The second inequality follows because $|S_{t+1}(x)| \leq |S_t(x)|$ for all $x \in \mathcal{X}$, and the final inequality follows once again from the fact that x^*_t maximizes $|S_t(x)|$. □

We next establish that, roughly speaking, δ_t must remain $O(\delta_1)$ for $\Omega(1/\delta_1)$ steps. More precisely:

Lemma 4.4. *If* $t < \frac{1}{\delta_1}$ *then* $\delta_t \leq \frac{\delta_1}{1-t\delta_1}$.

Proof. The base case $t = 1$ clearly holds. Now suppose for induction that $\delta_t \leq \frac{\delta_1}{1-t\delta_1}$. We have

$$\begin{aligned}\delta_{t+1} &\leq \frac{\delta_t}{1-\delta_t} \leq \frac{\delta_1}{1-t\delta_1}\left(\frac{1}{1-\delta_t}\right) = \frac{\delta_1}{1-t\delta_1-\delta_t+t\delta_1\delta_t} \\ &= \frac{\delta_1}{1-(t+1)\delta_1+\delta_1-\delta_t+t\delta_1\delta_t} \leq \frac{\delta_1}{1-(t+1)c\delta_1}\end{aligned}$$

The first inequality holds by Lemma 4.3, and the second inequality holds by the induction hypothesis. For the final inequality, note that $\delta_t \leq \frac{\delta_1}{1-t\delta_1}$ implies that $\delta_1 - \delta_t + t\delta_1\delta_t \geq 0$, as long as $t < \frac{1}{\delta_1}$. □

We are now ready to prove the lower bound of Theorem 4.2.

Proof. (Theorem 4.2) The behavior of algorithm A is partly determined by its internal randomization, which we denote as a random string ω drawn from a distribution $\mathcal{P}$. Let us write $A(\omega)$ for the deterministic algorithm corresponding to the string ω.

Fix any positive constant $c < 1/2$ (implying $\frac{c}{1-c} < 1$). For any fixed ω

$$\Pr_{f^* \sim U_{F_\theta}}\left[T^{A(\omega),f^*} < \frac{c}{\theta}\right] \leq \sum_{t=1}^{c/\theta}\delta_t \leq \sum_{t=1}^{c/\theta}\frac{\delta_1}{1-c} = \sum_{t=1}^{c/\theta}\frac{\theta}{1-c} = \frac{c}{1-c} \tag{4.1}$$

where we used, in order: Lemma 4.2; Lemma 4.4 and the fact that $c < 1$; $\theta = \delta_1$ (by definition); arithmetic. Now we have

$$E_{f^* \sim U_{F_\theta}}\left[T^{A(\omega),f^*}\right] \geq \left(1 - \Pr_{f^* \sim U_{F_\theta}}\left[T^{A(\omega),f^*} < \frac{c}{\theta}\right]\right)\frac{c}{\theta} \geq \left(1 - \frac{c}{1-c}\right)\frac{c}{\theta} \tag{4.2}$$

where the second inequality follows from (4.1). Finally, we have

$$E_{f^* \sim U_{F_\theta}}\left[T^{A,f^*}\right] \triangleq E_{\omega \sim \mathcal{P}, f^* \sim U_{F_\theta}}\left[T^{A(\omega),f^*}\right]$$
$$= E_{\omega \sim \mathcal{P}}\left[E_{f^* \sim U_{F_\theta}}\left[T^{A(\omega),f^*}|\omega\right]\right] \geq E_{\omega \sim \mathcal{P}}\left[\left(1 - \frac{c}{1-c}\right)\frac{c}{\theta}\right] = \left(1 - \frac{c}{1-c}\right)\frac{c}{\theta}$$

where the inequality follows from (4.2). The choice of c implies $\left(1 - \frac{c}{1-c}\right) c > 0$, which together with the definition $\mathrm{HD}(\mathcal{F}) \triangleq \frac{1}{\theta}$ implies the theorem, with $c = 2 - \sqrt{3}$ giving the best bound. □

4.7. Relationship to VC Dimension and Extended Teaching Dimension

As we have demonstrated, the haystack dimension provides a lower bound on the query complexity of any algorithm for the MAX problem on a function class $\mathcal{F}$. This is a role analogous to the VC dimension in the PAC learning model. However, as we will demonstrate, the two are incomparable in general. We will also illustrate the haystack dimension's relationship to the *extended teaching dimension* (Hegedüs, 1995). The extended teaching dimension characterizes the number of queries required to learn $f^* \in \mathcal{F}$ when $\mathcal{F}$ consists of binary functions. Clearly learning f^* is sufficient for maximization and, as we will see, the haystack dimension can be much smaller than extended teaching dimension, but cannot be too much larger. Note that the VC and extended teaching dimensions are defined only for binary functions, whereas the haystack dimension and our results encompass a much more general setting.

For $\mathcal{F}$ consisting of binary functions, we will denote the VC dimension as $VCD(\mathcal{F})$, where the hypothesis class is assumed to equal to concept class. Similarly, we will denote the extended teaching dimension by $XTD(\mathcal{F})$, redefined below.

Definition 4.2 (Hegedüs (1995))**.** *Let $h : \mathcal{X} \to \{0, 1\}$. We say that $S \subseteq \mathcal{X}$ is a* specifying set *for h if there is at most one $f \in \mathcal{F}$ consistent with h on all $x \in S$. Let $XTD(\mathcal{F}, h)$ be equal to the size of the smallest specifying set for h. $XTD(\mathcal{F}) = \sup_h XTD(\mathcal{F}, h)$.*

Example 4.5. *(a) For any d, there exists a function class and set of inputs $(\mathcal{F}, \mathcal{X})$ such that $VCD(\mathcal{F}) = d$ but $HD(\mathcal{F}) = 1$. (b) For any d, there exists a $(\mathcal{F}, \mathcal{X})$ such that $VCD(\mathcal{F}) = 2$ but $HD(\mathcal{F}) = d$.*

Proof. To prove (a), let $(\mathcal{F}_d, \mathcal{X}_d)$ be any function class, and set of inputs, such that $VCD(\mathcal{F}_d) = d$. Let $\mathcal{X} = \mathcal{X}_d \cup \{x^*\}$. Construct $(\mathcal{F}, \mathcal{X})$ by including a function f in $\mathcal{F}$ for each $f' \in \mathcal{F}_d$, that is identical to f' on all inputs in $\mathcal{X}_d$. Also let $f(x^*) = 1$. For any $F \subseteq \mathcal{F}$, x^* maximizes all functions in F. Therefore, $HD(\mathcal{F}) = 1$. However, no function in $\mathcal{F}$ gives x^* the label 0, and so the size of the largest shattered set does not change, and $VCD(\mathcal{F}) = d$.

To prove (b), consider $\mathcal{F}$ to be the "needle in the haystack" of Example 4.1, where $|\mathcal{F}| = d$. We have that $HD(\mathcal{F}) = d$. To compute the VC dimension, note that no function in $\mathcal{F}$ labels more than one input with a 1, and so no set of three inputs can be shattered by $\mathcal{F}$. (It's also obvious that any $\{x_1, x_2\} \subseteq \mathcal{X}$ can be shattered). □

Example 4.6. *(a) For any d, there exists a function class and set of inputs $(\mathcal{F}, \mathcal{X})$ such that $XTD(\mathcal{F}) = d$ but $HD(\mathcal{F}) = 1$. (b) For any $\mathcal{F}$ consisting of binary functions, $HD(\mathcal{F}) = O(XTD(\mathcal{F}) \log |\mathcal{F}|)$.*

Proof. To prove (a), let $(\mathcal{F}_d, \mathcal{X}_d)$ be a function class, and set of inputs, such that $XTD(\mathcal{F}_d) = d$. Construct $(\mathcal{F}, \mathcal{X})$ exactly as in the proof of Example 4.5(a). For any $h : \mathcal{X} \to \{0, 1\}$ with $h(x^*) = 1$, let $h' : \mathcal{X} \to \{0, 1\}$ be a function identical to h on $\mathcal{X}_d$. It's clear that $XTD(\mathcal{F}, h) = XTD(\mathcal{F}_d, h')$. For any $h : \mathcal{X} \to \{0, 1\}$ with $h(x^*) = 0$, $XTD(\mathcal{F}, h) = 1$, since $\{x^*\}$ is a specifying set for h. Thus, $XTD(\mathcal{F}) = XTD(\mathcal{F}_d) = d$.

For (b), Hegedüs (1995) gives an algorithm which learns $f^* \in \mathcal{F}$ using $O(XTD(\mathcal{F}) \log |\mathcal{F}|)$ membership queries. Since learning f^* is sufficient for maximization, Theorem 4.2 implies (b). □

4.8. Functional MAB Regret Upper Bound

In this and the next section, we return to the functional MAB problem, showing a close relationship to the functional MAX problem, and giving upper and lower bounds on regret that are order the haystack dimension $\mathrm{HD}(\mathcal{F})$. We will describe a no-regret algorithm for functional MAB problems where $\mathcal{F}$ is known, finite, but otherwise arbitrary. Our approach is to use the greedy functional MAX algorithm of Section 4.5 to find an action/query that maximizes f^*, and then select that action indefinitely. There is a technical complication we must overcome when implementing this approach: Each action returns a sample from a distribution, rather than a single value. We solve this by repeatedly selecting an action x to get an accurate estimate of $f^*(x)$, and also by restarting the algorithm after progressively longer phases (a standard "trick" in MAB algorithms).

Before giving a more detailed description of our algorithm and its analysis, we need some additional definitions. Let $\epsilon_{\min} \triangleq \min_{f,f' \in \mathcal{F}} \inf_{x:f(x)\neq f'(x)} |f(x) - f'(x)|$ be the smallest difference between any two functions in $\mathcal{F}$ on those points where they do differ; $\epsilon_{\min} > 0$ allows us to determine $f^*(x)$ by selecting x enough times. Also, for any set $F \subseteq \mathcal{F}$, let us define the *ϵ-inconsistent set* to be $F(\langle x, y\rangle, \epsilon) \triangleq \{f \in F : |f(x) - y| > \epsilon\}$.

The *greedy bandit algorithm GB* proceeds in phases $i = 1, 2, \ldots$, where each phase lasts $T_i = 2^{2^i}$ rounds. Each phase consists of two interleaved 'threads' of execution. That is, each thread specifies a sequence of actions, and actions are alternatingly selected from each thread. The 'explore' thread runs a fresh instance of the greedy query algorithm G from Section 4.5, which selects actions $x_1^i, x_2^i, \ldots$ until it empties the attention space. We make one minor modification to the greedy algorithm G; the attention space F_t^i maintained by G during the explore thread of phase i is updated according to the rule $F_{t+1}^i = F_t^i \setminus \left(F_t^i(x_t^i) \cup F_t^i\left(\left\langle x_t^i, \bar{y}_t^i\right\rangle, \frac{\epsilon_{\min}}{2}\right)\right)$, where $\bar{y}_t^i$ is an estimate of $f^*(x_t^i)$ obtained by selecting action x_t^i several times and averaging the observations. As we shall see in the proof of Theorem 4.3, despite this change, we will still be able to bound the number of actions selected by this modified version of G in terms of the haystack dimension $\mathrm{HD}(\mathcal{F})$. The 'exploit' thread

of phase i always selects the action $x^i_{t^*}$ associated with the largest average observation $\bar{y}^i_{t^*}$ in this phase. Phase i terminates as soon as the total number of action selected by both threads reaches T_i. Pseudocode is given in Algorithm 5 below.

Theorem 4.3. *Let GB be the greedy bandit algorithm for the MAB problem for $\mathcal{F}$ (see Algorithm 5). Then*

$$R_{GB}(T) = O\left(\frac{\mathrm{HD}(\mathcal{F})}{\epsilon^2_{\min}} \log|\mathcal{F}| \left(\log T + \log(\mathrm{HD}(\mathcal{F})) + \log\log|\mathcal{F}|\right) + \log\log T\right)$$

Proof. Let R_i be the realized regret during phase i. We will first bound $E[R_i]$ for any phase i, and then use a 'squaring trick' to tightly bound $E[\sum_i R_i] \triangleq R(T)$.

For each phase i, let good(i) be the event that $|\bar{y}^i_t - f^*(x^i_t)| \leq \frac{\epsilon_{\min}}{2}$ for all $t \leq \mathrm{HD}(\mathcal{F}) \log|\mathcal{F}|$. In the explore thread, each action x^i_t is selected $O\left(\frac{1}{\epsilon^2_{\min}} \left(\log T_i + \log(\mathrm{HD}(\mathcal{F})) + \log\log|\mathcal{F}|\right)\right)$ times, which is enough to ensure (by the Chernoff and union bounds) that the event good(i) occurs with probability $1 - O(\frac{1}{T_i})$ (see Algorithm 5). Thus we have

$$\begin{aligned} E[R_i] &= E[R_i|\,\mathrm{good}(i)]\,\Pr[\mathrm{good}(i)] + E[R_i|\neg\,\mathrm{good}(i)]\,\Pr[\neg\,\mathrm{good}(i)] \\ &\leq E[R_i|\,\mathrm{good}(i)] + O(1) \end{aligned} \tag{4.3}$$

because $R_i \leq T_i$ and $\Pr[\neg\,\mathrm{good}(i)] \leq O(\frac{1}{T_i})$. It remains to bound $E[R_i|\,\mathrm{good}(i)]$.

By the definition of $\epsilon_{\min}$, if the event good(i) occurs then the attention space F^i_t maintained in the explore thread is updated exactly the same way as in the greedy query algorithm G from Section 4.5. Therefore, the explore thread eventually selects an action in X^{f^*}, and thus, by the definition of $\epsilon_{\min}$, the exploit thread selects an action in X^{f^*} every round after the explore thread ends. Since the explore thread ends after $O\left(\frac{\tau_i}{\epsilon^2_{\min}} \left(\log T_i + \log(\mathrm{HD}(\mathcal{F})) + \log\log|\mathcal{F}|\right)\right)$ rounds, and $\tau_i \leq \mathrm{HD}(\mathcal{F}) \log|\mathcal{F}|$ by Theorem 4.1, we have

$$E[R_i|\,\mathrm{good}(i)] \leq O\left(\frac{\mathrm{HD}(\mathcal{F})}{\epsilon^2_{\min}} \log|\mathcal{F}| \left(\log T_i + \log(\mathrm{HD}(\mathcal{F})) + \log\log|\mathcal{F}|\right)\right) \tag{4.4}$$

Finally, we apply a 'squaring trick'.[5] Let K be the number of phases. Since $T_i = 2^{2^i}$, we have $K \leq O(\log\log T)$ and

$$\sum_{i=1}^{K} \log T_i \leq \sum_{i=1}^{O(\log\log T)} 2^i = O(\log T) \tag{4.5}$$

Combining (4.3), (4.4), and (4.5), and recalling that $R(T) \triangleq E[\sum_i R_i]$, proves the theorem.

□

Algorithm 5 Greedy Bandit Algorithm (GB)

for $i = 1, 2, \ldots$ **do**
  for $T_i = 2^{2^i}$ rounds **do**
    Interleave the following two threads:

    Explore Thread:
    Let $F_1^i = \mathcal{F}$ and $\tau_i = 1$.
    while $F_{\tau_i}^i$ is non-empty **do**
      Let $x_{\tau_i}^i = \arg\sup_{x\in\mathcal{X}} \inf_y |F_{\tau_i}^i(x) \cup F_{\tau_i}^i(\langle x, y\rangle, \frac{\epsilon_{\min}}{2})|$
      Select query $x_{\tau_i}^i$ for $O\left(\frac{1}{\epsilon_{\min}^2}\left(\log T_i + \log\frac{1}{\theta} + \log\log|\mathcal{F}|\right)\right)$ rounds, and let $\bar{y}_{\tau_i}^i$ be the average observation.
      Let $F_{\tau_i+1}^i = F_{\tau_i}^i \setminus (F_{\tau_i}^i(x_{\tau_i}^i) \cup F_{\tau_i}^i(\langle x_{\tau_i}^i, \bar{y}_{\tau_i}^i\rangle, \epsilon_{\min}))$
      Let $\tau_i = \tau_i + 1$.
    end while

    Exploit Thread:
    Let $f_t^i = \arg\min_{f\in\mathcal{F}} |f(x_t^i) - \bar{y}_t^i|$ for all $t \leq \tau_i$.
    Let $\hat{y}_t^i = f_t^i(x_t^i)$ for all $t \leq \tau_i$.
    Select query $x_{t^*}^i$ each round, where $t^* = \arg\sup_{1\leq t\leq\tau} \hat{y}_t^i$.
  end for
end for

Remark: Note that the greedy bandit algorithm GB assumes that the value of the haystack dimension $\mathrm{HD}(\mathcal{F})$ is known. It is possible to modify the algorithm in case this information is not available: In Algorithm 5, each action x_t^i selected in the explore thread of phase i is selected $O\left(\frac{1}{\epsilon_{\min}^2}\left(\log T_i + \log(\mathrm{HD}(\mathcal{F})) + \log\log|\mathcal{F}|\right)\right)$ times. If $\mathrm{HD}(\mathcal{F})$ is unknown, we simply change this to $O\left(\frac{1}{\epsilon_{\min}^2}\left(\log T_i + \log|\mathcal{F}| + \log\log|\mathcal{F}|\right)\right)$ times. Since $\mathrm{HD}(\mathcal{F}) \leq |\mathcal{F}|$ trivially holds, the analysis in the proof of Theorem 4.3 is essentially unaffected by this modification, and it adds only a $O((\log|\mathcal{F}|)^2)$ term to the regret upper bound in Theorem

[5] If we were to instead apply the more common 'doubling trick', such that $T_i = 2^i$, the upper bound in Eq. (4.5) would be $O((\log T)^2)$.

4.3.

4.9. Functional MAB Regret Lower Bound

In this section, we prove that the greedy bandit algorithm in Section 4.8 is near-optimal, at least with respect to the haystack dimension (our primary interest) and terms $\log|\mathcal{F}|$ or smaller. With respect to the dependence on the haystack dimension, we can say something quite strong: Every MAB algorithm for every function class must suffer regret that is linear in the haystack dimension of the class. Let $\Delta = \inf_{f\in\mathcal{F}} \inf_{\{x'\in\mathcal{X}:f(x')<\sup_x f(x)\}} \sup_x f(x) - f(x)$, be the difference between the best action and second-best action in $\mathcal{X}$.

Theorem 4.4. *For any function class $\mathcal{F}$ and MAB algorithm A for $\mathcal{F}$:*

$$R_A(T) = \Omega\left(\Delta \min\{T, \mathrm{HD}(\mathcal{F})\}\right)$$

The proof of Theorem 4.4 follows directly from Theorem 4.2, which implies that no MAB algorithm for function class $\mathcal{F}$ can select the best action in fewer than $\mathrm{HD}(\mathcal{F})$ steps. The next theorem proves that the terms $\log|\mathcal{F}|$ and $\frac{1}{\epsilon_{\min}}$ cannot be removed from the upper bound in Theorem 4.3.

Theorem 4.5. *(a) There exists a function class $\mathcal{F}$ such that $\mathrm{HD}(\mathcal{F}) = \Theta(1)$ and for any MAB algorithm A for $\mathcal{F}$, $R_A(T) = \Omega\left(\log|\mathcal{F}|\right)$. (b) There exists a function class $\mathcal{F}$ such that $\mathrm{HD}(\mathcal{F}) = \Theta(1)$ and for any MAB algorithm A for $\mathcal{F}$, $R_A(T) = \Omega\left(\min\left\{\frac{1}{\epsilon_{\min}}, |\mathcal{F}|\right\}\right)$.*

Proof. To prove (a), we will outline the construction of $\mathcal{F}$ and omit the details of the proof, which are straightforward. The input space $\mathcal{X}$ will have two components — $\mathcal{X}_1$ and $\mathcal{X}_2$, with $\mathcal{X}$ being the union of these disjoint domains. Subdomain $\mathcal{X}_1$ consists of 2^n points, and $\mathcal{X}_2$ of n points, while the function class $\mathcal{F}$ contains n deterministic functions. Each point in $x \in \mathcal{X}_1$ corresponds to a distinct subset $F_x \subseteq \mathcal{F}$, and for each $x \in \mathcal{X}_1$ let $f(x) = \frac{1}{3}$ for half the functions in F_x and $f(x) = \frac{2}{3}$ for the other half, and also let $f(x) = \frac{1}{3}$ for all $f \in \mathcal{F} \setminus F_x$. Note that $\mathcal{X}_1$ contains excellent information queries, because for any subset $F \subseteq \mathcal{F}$ there

is a point in $\mathcal{X}_1$ that eliminates at least half the functions in F when that point is issued as a query, and thus $\mathrm{HD}(\mathcal{F}) = 2$. Finally, map each function $f \in \mathcal{F}$ to a distinct point $x_f \in \mathcal{X}_2$, and let $f(x_f) = 1$ and $f(x) = 0$ for all $x \in \mathcal{X}_2 \setminus \{x_f\}$. So each $f \in \mathcal{F}$ has a unique maximum, contained in $\mathcal{X}_2$. Suppose f^* is chosen uniformly at random from $\mathcal{F}$. It is clear that $Q_A = \Omega(\log |\mathcal{F}|)$. To see why, note that a query $x \in \mathcal{X}_2$ has only probability $\frac{1}{|\mathcal{F}|}$ of being the function's max, and only serves to inform the algorithm that $f^* \neq f$ whenever $x = x_f$ and $f^*(x) = 0$. Thus any algorithm is forced to learn f^* by playing actions in $\mathcal{X}_1$, requiring that at least $\log_2 |\mathcal{F}|$ actions are played.

To prove (b), we simply modify the construction of $\mathcal{F}$ to add stochasticity, as follows. For an $x \in \mathcal{X}_1$, if we had previously that $f(x) = \frac{1}{3}$, we now let $f(x) = \epsilon$. If we had previously that $f(x) = \frac{2}{3}$, we now let $f(x) = 2\epsilon$. Each function's behavior of $\mathcal{X}_2$ is unchanged. Note that $\epsilon_{\min} = \epsilon$. When playing an $x \in \mathcal{X}$, rather than observe the output of the function, the algorithm now observes the outcome a Bernoulli random variable with mean $f^*(x)$. It can then be shown that any algorithm that wishes to make use of the information in $\mathcal{X}_1$ must sample the actions there $\Omega(1/\epsilon)$ times or else be forced to play actions in $\mathcal{X}_2$. □

4.10. Infinite Function Classes

In the remainder of the chapter, we return to studying the MAX problem. We give extensions of our basic results on query complexity, and also give examples that illustrate some aspects of our analysis. In this section, we describe how to extend our methods from Sections 4.5 and 4.6, which were restricted to finite function classes, to infinite function classes that have a *finite covering oracle*.

Definition 4.3. *We say C is a* finite covering oracle *for $\mathcal{F}$ if for any $\epsilon > 0$ the finite set $C(\epsilon) \subseteq \mathcal{F}$ has the following property: For any $f \in \mathcal{F}$ there exists an $f' \in C(\epsilon)$ such that* $\sup_{x \in \mathcal{X}} |f(x) - f'(x)| \leq \epsilon$.

Fix a possibly infinite function class $\mathcal{F}$ with finite covering oracle C. We consider the ϵ-MAX problem for $\mathcal{F}$, a relaxed version of the MAX problem. In analogy to the MAX problem,

for any algorithm A let $T^{A,f^*,\epsilon} \triangleq \min\{t : \sup_x f^*(x) - f^*(x_t) \leq \epsilon\}$ be the first round t such that the query x_t selected by A is an *ϵ-maximum* of $f^* \in \mathcal{F}$. We are interested in bounding the *worst-case ϵ-query complexity* of A, defined $Q_{A,\epsilon} \triangleq \sup_{f^* \in \mathcal{F}} E[T^{A,f^*,\epsilon}]$.

Below we give upper and lower bounds for the ϵ-MAX problem in terms of the *lower and upper ϵ-haystack dimensions*, which are each a different generalization of the haystack dimension. Before introducing these quantities, we need some definitions. For any set $F \subseteq \mathcal{F}$ let $F(x,\epsilon) \triangleq \{f \in F : \sup_{x'} f(x') - f(x) \leq \epsilon\}$ be the functions in F for which x is an ϵ-maximum and, as in Section 4.8, let $F(\langle x,y\rangle,\epsilon) \triangleq \{f \in F : |f(x) - y| > \epsilon\}$ be the functions in F that are more than ϵ-inconsistent with the labeled example $\langle x,y\rangle$. Also, let

$$\theta^-(\epsilon) \triangleq \inf_{F \subseteq C(\epsilon)} \sup_{x \in \mathcal{X}} \inf_y \frac{|F(x,\epsilon) \cup F(\langle x,y\rangle,0)|}{|F|} \quad \text{and} \quad \theta^+(\epsilon) \triangleq \inf_{F \subseteq C(\epsilon)} \sup_{x \in \mathcal{X}} \inf_y \frac{|F(x,0) \cup F(\langle x,y\rangle,\epsilon)|}{|F|}$$

Note that the only difference between $\theta^-(\epsilon)$ and $\theta^+(\epsilon)$ is the placement of ϵ and 0. Also note that if $\mathcal{F}$ is finite and $C(\epsilon) = \mathcal{F}$ then $\theta^-(0) = \theta^+(0) = \theta$, where θ was defined in Section 4.3. Now define the *lower ϵ-haystack dimension* $\mathrm{HD}^-(\epsilon) \triangleq \frac{1}{\theta^-(\epsilon)}$ and the *upper ϵ-haystack dimension* $\mathrm{HD}^+(\epsilon) \triangleq \frac{1}{\theta^+(\epsilon)}$.

A simple approach to solving the ϵ-MAX problem is just to run a slightly modified version of the greedy algorithm from Section 4.5 using $C(\epsilon)$ as the initial attention space, and removing inconsistent functions only if they are inconsistent by more than ϵ. In other words, in each round t, the *ϵ-greedy algorithm* G_ϵ selects $x_t = \arg\sup_{x \in \mathcal{X}} \inf_y |F_t(x,0) \cup F_t(\langle x,y\rangle,\epsilon)|$ where the *attention space* F_t is defined inductively as follows: $F_1 = C(\epsilon)$ and $F_{t+1} = F_t \setminus (F_t(x_t,0) \cup F_t(\langle x_t,y_t\rangle,\epsilon))$.

Theorem 4.6. *Let G_ϵ be the ϵ-greedy algorithm for the ϵ-MAX problem. Then $Q_{G_\epsilon,2\epsilon} \leq \mathrm{HD}^+(\epsilon) \log |C(\epsilon)|$ for all $\epsilon > 0$.*

Proof. The proof is nearly identical to the proof of Theorem 4.1. The algorithm G_ϵ initializes the attention space to $C(\epsilon)$, and after every query at least a $\theta^+(\epsilon)$ fraction of the attention space is eliminated. By the time the attention space is empty, the G_ϵ algorithm has selected a maximum of some function $\hat{f} \in C(\epsilon)$ that ϵ-covers the true function f^*, which implies

that a 2ϵ-maximum of f^* has been selected. $\square$

Furthermore, we can also straightforwardly lower bound the query complexity of any algorithm for the ϵ-MAX problem.

Theorem 4.7. *Let A be any algorithm for the ϵ-MAX problem. Then $Q_{A,\epsilon} = \Omega\left(\mathrm{HD}^-(\epsilon)\right)$ for all $\epsilon > 0$.*

Proof. The proof of Theorem 4.2 can be repeated, with essentially no changes, to prove this theorem as well. The key is to observe that, when proving Theorem 4.2, we made no use of the fact that X^{f^*} contained the maxima of the true function f^*. We could have defined X^{f^*} to be any subset of $\mathcal{X}$, including the ϵ-maxima of f^*. $\square$

Notice that the upper and lower bounds in Theorems 4.6 and 4.7 are not directly comparable, since we have not related the quantities $\mathrm{HD}^-(\epsilon)$ and $\mathrm{HD}^+(\epsilon)$. If it happens that $\frac{\mathrm{HD}^+(\epsilon)}{\mathrm{HD}^-(\epsilon)} \leq K$ for some constant K, then clearly the upper and lower bounds above are within constant and logarithmic factors of each other, just as we had for finite function classes. Indeed, a simple infinite function class for which this occurs is the class of all bounded norm hyperplanes. Let $\mathcal{X} = \{x \in \mathbb{R}^n \mid \|x\|_\infty \leq 1\}$ and $\mathcal{F}_{\text{hyper}} = \{\langle w, \cdot\rangle \mid w \in \mathbb{R}^n, \|w\|_\infty \leq 1\}$, and let the finite covering oracle C be the appropriate discretization of $\mathcal{F}_{\text{hyper}}$, i.e., $C(\epsilon) = \{w \in \mathbb{R}^n \mid \forall i\ w_i \in \{0, \frac{\epsilon}{n}, \frac{2\epsilon}{n}, \ldots, 1\}\}$. Clearly $|C(\epsilon)| = \Theta((\frac{n}{\epsilon})^n)$, but nonetheless the ratio of the lower and upper ϵ-haystack dimension is not large.

Theorem 4.8. *For function class $\mathcal{F}_{\text{hyper}}$ we have $\frac{\mathrm{HD}^+(\epsilon)}{\mathrm{HD}^-(\epsilon)} \leq n$ for all $\epsilon > 0$.*

Proof. By examining the definitions of $\mathrm{HD}^+(\epsilon)$ and $\mathrm{HD}^-(\epsilon)$, we see that it suffices to show that $\sup_{x \in \mathcal{X}} \frac{|F(x,0)|}{|F|} \geq \frac{1}{n}$ for any finite $F \subseteq \mathcal{F}$. Let $k_i = |\{\langle w, \cdot\rangle \in F \mid w_j \geq w_i, \forall j\}|$ be the number of functions in F which have their maximal component at i. Clearly there must exist a $k_i \geq \frac{|F|}{n}$. Let e_i be the vector with a one in its ith component and zeros everywhere else. Since $e_i \in \mathcal{X}$ and e_i maximizes k_i functions in F, there exists an $x \in \mathcal{X}$ with $\frac{|F(x,0)|}{|F|} \geq \frac{1}{n}$. $\square$

Unfortunately, the bound in Theorem 4.8 cannot be generalized to all function classes with finite covering oracles. In the next theorem, we give an example of an infinite function class $\mathcal{F}$ with a finite covering oracle C for which $\frac{\mathrm{HD}^+(\epsilon)}{\mathrm{HD}^-(\epsilon)} = \Omega(|C(\epsilon)|)$, which is essentially the worst possible ratio.

Theorem 4.9. *There exists a function class $\mathcal{F}$ with a finite covering oracle C such that $\mathrm{HD}^-(\epsilon) = 1$ and $\mathrm{HD}^+(\epsilon) = \Omega(|C(\epsilon)|)$ for all $\epsilon > 0$.*

Proof. Let $\mathcal{X} = [0, 1]$ and fix any sequence $\{x_n\} \subset \mathcal{X}$, all elements distinct. For visualization, it may be helpful to suppose that $\{x_n\}$ is strictly increasing, but this is not necessary. Let $\{\epsilon_i\}$ be the sequence defined by $\epsilon_i = \frac{1}{2^i}$ for all $i \in \mathbb{N}$.

For each $n \in \mathbb{N}$ let there be a function $f_n \in \mathcal{F}$ whose values are zero everywhere except $x_1, \ldots, x_n$. The nonzero values of f_n are defined as follows: Let $f_n(x_n) = \frac{1}{n}$ and $f_n(x_m) = \epsilon_i - \frac{1}{2^{2^n}}$ for all $m < n$, where $i = \lceil \log_2 n \rceil$. Let $C(\epsilon) = \{f_1, \ldots, f_{N_\epsilon}\}$, where N_ϵ is the smallest integer such that $1/N_\epsilon < \epsilon$. We have that C is finite covering oracle because each $C(\epsilon)$ is finite and because $\sup_{x \in \mathcal{X}} f_n(x) \leq \epsilon$ for all $n \geq N_\epsilon$.

We only need the following properties of this construction, which can be verified: (1) For all $n \in \mathbb{N}$ the elements of $\{f_m(x_n) : m \geq n\}$ are all distinct; (2) Each f_n has a maximum at x_n and nowhere else; (3) For all $n \in \mathbb{N}$, if $i = \lceil \log_2 n \rceil$ then $f_n(x_n) \geq \epsilon_i$ and $f_m(x_n) < \epsilon_i$ for all $m \neq n$.

For the remainder of the proof fix $\epsilon > 0$ and the smallest $i \in \mathbb{N}$ such that $\epsilon_i \leq \epsilon$. We will show that $\theta^-(\epsilon) = 1$ and $\theta^+(\epsilon) \leq \frac{4}{N_{\epsilon_i}}$, which suffices to prove the theorem.

First, we claim that $\theta^-(\epsilon) = 1$. Let $F^+ = \arg\inf_{F \subseteq C(\epsilon)} \sup_{x \in \mathcal{X}} \inf_y \frac{|F(x,\epsilon) \cup F(\langle x, y \rangle, 0)|}{|F|}$. Let n be the smallest integer such that $f_n \in F^+$. Then each $f_m \in F^+$ has a distinct value at x_n, by property 1. So $\inf_y |F^+(\langle x_n, y \rangle, 0)| = |F^+|$. Next, we claim that $\theta^+(\epsilon) \leq \frac{4}{N_{\epsilon_i}}$. Let $F_i = \{f_n \in \mathcal{F} : i = \lceil \log_2 n \rceil\}$, and note that $F_i \subseteq C(\epsilon)$ and $|F_i| = \frac{N_{\epsilon_i}}{2}$. For any $x \in \mathcal{X}$ we have $|F_i(x, 0)| \leq 1$, by property 2, and $\inf_y |F_i(\langle x, y \rangle, \epsilon)| \leq 1$, by property 3. □

4.11. Computational Complexity

Our results have so far ignored computational complexity. In general a function class $\mathcal{F}$, for which finding the optimal query is computationally intractable, might nevertheless have small haystack dimension, admitting algorithms with low query complexity. Consider the following simple example. Let $\mathcal{X} = \mathcal{X}_1 \cup \mathcal{X}_2 \cup \mathcal{X}_3$ where $\mathcal{X}_1, \mathcal{X}_2, \mathcal{X}_3$ are disjoint, $|\mathcal{X}_1| = |\mathcal{X}_2| = n$ and $\mathcal{X}_3 = 2^n$. Each function in $\mathcal{F}$ will attain its maximum on some action in $\mathcal{X}_3$. The location of that maximum, as in Example 4.2, will be encoded by $\mathcal{X}_1$ and $\mathcal{X}_2$. However, we will now encode the location cryptographically, with a function's behavior on $\mathcal{X}_1$ representing a public key, and a function's behavior on $\mathcal{X}_2$ representing the encrypted location of its maximum.

More precisely, let z be an n-bit number. For a pair of $\frac{n}{2}$-bit primes p and q, let $N_{pq} = pq$. Also let $e(z, N_{pq}) = z^2 \mod N_{pq}$ be z encrypted with public key N_{pq}.

Now let $f_{z,p,q}$ be the function that gives the ith bit of N_{pq} as output to the ith input of $\mathcal{X}_1$ and the ith bit of $e(z, N_{pq})$ as output to the ith input of $\mathcal{X}_2$. On the zth input of $\mathcal{X}_3$, $f_{z,p,q}$ outputs a 2. On all other inputs of $\mathcal{X}_3$, $f_{z,p,q}$ outputs 0. Let $\mathcal{F}$ be the function class consisting of all functions $f_{z,p,q}$ for every pair of $\frac{n}{2}$-bit primes p, q and n-bit integer z.

There exists an algorithm with query complexity $O(n)$ for any $f^* \in \mathcal{F}$. That algorithm queries each action in $\mathcal{X}_1 \cup \mathcal{X}_2$, retrieving the public key N_{pq} and the cypher $e(z, N_{pq})$. Information-theoretically, the maximum of f^* can be found in a single additional query. The algorithm may simply test every n-bit z', checking if $e(z, N_{pq}) = e(z', N_{pq})$. However, computing z is as hard as factorization (Kearns and Vazirani, 1994).

CHAPTER 5 : Posted Price Auctions Against Strategic Buyers

Our models so far have not required any *strategic* reasoning; we have not considered the possibility that the market would react to our choice of algorithm. In many settings, this is quite reasonable. When making decisions in a crowded market, although individual agents act strategically, the macroscopic market is unaffected by an individual firm's decisions, and adequately modeled as a stochastic function. In this chapter, we consider the opposite extreme, motivated by an application from real-time bidding on ad exchanges.

Online display advertising inventory — e.g., space for banner ads on web pages — is often sold via automated transactions on real-time ad exchanges. When a user visits a web page whose advertising inventory is managed by an ad exchange, a description of the web page, the user, and other relevant properties of the *impression*, along with a *reserve price* for the impression, is transmitted to bidding servers operating on behalf of advertisers. These servers process the data about the impression and respond to the exchange with a bid. The highest bidder wins the right to display an advertisement on the web page to the user, provided that the bid is above the reserve price. The amount charged the winner, if there is one, is settled according to a second-price auction. The winner is charged the maximum of the second-highest bid and the reserve price.

Ad exchanges have been a boon for advertisers, since rich and real-time data about impressions allow them to target their bids to only those impressions that they value. However, this precise targeting has an unfortunate side effect for web page publishers. A nontrivial fraction of ad exchange auctions involve only a *single* bidder. Without competitive pressure from other bidders, the task of maximizing the publisher's revenue falls entirely to the reserve price setting mechanism. Second-price auctions with a single bidder are equivalent to *posted-price* auctions. The seller offers a price for a good, and a buyer decides whether to accept or reject the price (i.e., whether to bid above or below the reserve price).

In this chapter we consider online learning algorithms for setting prices in posted-price

auctions where the seller repeatedly interacts with the *same* buyer over a number of rounds, a common occurrence in ad exchanges where the same buyer might be interested in buying thousands of user impressions daily. In each round t, the seller offers a good to a buyer for price p_t. The buyer's value v_t for the good is drawn independently from a fixed value distribution. Both v_t and the value distribution are known to the buyer, but neither is observed by the seller. If the buyer accepts price p_t, the seller receives revenue p_t, and the buyer receives *surplus* $v_t - p_t$. Since the same buyer participates in the auction in each round, the seller has the opportunity to *learn* about the buyer's value distribution and set prices accordingly. Notice that in worst-case repeated auctions there is no such opportunity to learn, while standard Bayesian auctions assume knowledge of a value distribution, but avoid addressing how or why the auctioneer was ever able to estimate this distribution.

Taken as an online learning problem, we can view this as a bandit problem since the revenue for any price not offered is not observed (e.g., even if a buyer rejects a price, she may well have accepted a lower price). The seller's goal is to maximize his expected revenue over all T rounds. One straightforward way for the seller to set prices would therefore be to use a *no-regret* bandit algorithm, which minimizes the difference between seller's revenue and the revenue that would have been earned by offering the best fixed price p^* in hindsight for all T rounds; for a no-regret algorithm (such as UCB Auer et al. (2002) or EXP3 Auer et al. (2003)), this difference is $o(T)$. However, we argue that traditional no-regret algorithms are inadequate for this problem. Consider the motivations of a buyer interacting with an ad exchange where the prices are set by a no-regret algorithm, and suppose for simplicity that the buyer has a fixed value $v_t = v$ for all t. The goal of the buyer is to acquire the most valuable advertising inventory for the least total cost, i.e., to maximize her total surplus $\sum_t v - p_t$, where the sum is over rounds where the buyer accepts the seller's price. A naive buyer might simply accept the seller's price p_t if and only if $v_t \geq p_t$; a buyer who behaves this way is called *truthful*. Against a truthful buyer any no-regret algorithm will eventually learn to offer prices $p_t \approx v$ on nearly all rounds. But a more savvy buyer will notice that if she rejects prices in earlier rounds, then she will tend to see lower prices in later

rounds. Indeed, suppose the buyer only accepts prices below some small amount ϵ. Then any no-regret algorithm will learn that offering prices above ϵ results in zero revenue, and will eventually offer prices below that threshold on nearly all rounds. In fact, the smaller the learner's regret, the faster this convergence occurs. If $v \gg \epsilon$ then the deceptive buyer strategy results in a large gain in total surplus for the buyer, and a large loss in total revenue for the seller, relative to the truthful buyer. While the no-regret guarantee certainly holds — in hindsight, the best price is indeed ϵ — it seems fairly useless.

We propose a definition of *strategic regret* that accounts for the buyer's incentives, and give algorithms that are no-regret with respect to this definition. In our setting, the seller chooses a learning algorithm for selecting prices and announces this algorithm to the buyer. We assume that the buyer will examine this algorithm and adopt whatever strategy maximizes her expected surplus over all T rounds. We define the seller's strategic regret to be the difference between his expected revenue and the expected revenue he would have earned if, rather than using his chosen algorithm to set prices, he had instead offered the best fixed price p^* on all rounds *and the buyer had been truthful*. As we have seen, this revenue can be much higher than the revenue of the best fixed price in hindsight (in the example above, $p^* = v$). Unless noted otherwise, throughout the remainder of the chapter the term "regret" will refer to strategic regret.

We make one further assumption about buyer behavior, which is based on the observation that in many important real-world markets — and particularly in online advertising — sellers are far more willing to wait for revenue than buyers are willing to wait for goods. For example, advertisers are often interested in showing ads to users who have recently viewed their products online (this practice is called 'retargeting'), and the value of these user impressions decays rapidly over time. Or consider an advertising campaign that is tied to a product launch. A user impression that is purchased long after the launch (such as the release of a movie) is almost worthless. To model this phenomenon we multiply the buyer's surplus in each round by a *discount factor*: If the buyer accepts the seller's price p_t

in round t, she receives surplus $\gamma_t(v_t - p_t)$, where $\{\gamma_t\}$ is a nonincreasing sequence contained in the interval $(0, 1]$. We call $T_\gamma = \sum_{t=1}^{T} \gamma_t$ the buyer's 'horizon', since it is analogous to the seller's horizon T. The buyer's horizon plays a central role in our analysis.

Summary of results: In Sections 5.3 and 5.4 we assume that discount rates decrease geometrically: $\gamma_t = \gamma^{t-1}$ for some $\gamma \in (0, 1]$. In Section 5.3 we consider the special case that the buyer has a fixed value $v_t = v$ for all rounds t, and give an algorithm with regret at most $O(T_\gamma \sqrt{T})$. In Section 5.4 we allow the v_t to be drawn from any distribution that satisfies a certain smoothness assumption, and give an algorithm with regret at most $\tilde{O}(T^\alpha + T_\gamma^{1/\alpha})$ where $\alpha \in (0, 1)$ is a user-selected parameter. Note that for either algorithm to be no-regret (i.e., for regret to be $o(T)$), we need that $T_\gamma = o(T)$. In Section 5.5 we prove that this requirement is necessary for no-regret: any seller algorithm has regret at least $\Omega(T_\gamma)$. The lower bound is proved via a reduction to a non-repeated, or 'single-shot', auction. That our regret bounds should depend so crucially on T_γ is foreshadowed by the example above, in which a deceptive buyer foregoes surplus in early rounds to obtain even more surplus is later rounds. A buyer with a short horizon T_γ will be unable to execute this strategy, as she will not be capable of bearing the short-term costs required to manipulate the seller.

We note that since the first publication of this work (Amin et al., 2013), there have been (Amin et al., 2014; Mohri and Munoz, 2014) developing more principled algorithms with better rates, including extensions to the contextual setting. Here we demonstrate simply that the goal of no strategic regret is simply possible.

5.1. Related work

Kleinberg and Leighton study a posted price repeated auction with goods sold sequentially to T bidders who either all have the same fixed private value, private values drawn from a fixed distribution, or private values that are chosen by an oblivious adversary (an adversary that acts independently of observed seller behavior) Kleinberg and Leighton (2003) (see also Bar-Yossef et al. (2002); Blum et al. (2003)). Cesa-Bianchi et al. study a related problem of

setting the reserve price in a second price auction with multiple (but not repeated) bidders at each round Cesa-Bianchi et al. (2013). Note that none of these previous works allow for the possibility of a strategic buyer, i.e. one that acts non-truthfully in order to maximize its surplus. This is because a new buyer is considered at each time step and if the seller behavior depends only on previous buyers, then the setting immediately becomes *strategyproof*.

Contrary to what is studied in these previous theoretical settings, electronic exchanges in practice see the same buyer appearing in multiple auctions and, thus, the buyer has incentive to act strategically. In fact, Edelman and Ostrovsky (2007) finds empirical evidence of buyers' strategic behavior in sponsored search auctions, which in turn negatively affects the seller's revenue. In the economics literature, 'intertemporal price discrimination' refers to the practice of using a buyer's past purchasing behavior to set future prices. Previous work Acquisti and Varian (2005); Fudenberg and Villas-Boas (2006) has shown, as we do in Section 5.5, that a seller cannot benefit from conditioning prices on past behavior if the buyer is not myopic and can respond strategically. However, in contrast to our work, these results assume that the seller knows the buyer's value distribution.

Repeated posted price actions against the same strategic buyer have been considered in the economics literature under the heading of *behavior-based price discrimination* (BBPD) by Hart and Tirole (1988); Schmidt (1993); Acquisti and Varian (2005); Fudenberg and Villas-Boas (2006), and more recently by Devanur et al. (2014). These works differ from ours in two key ways. First, all these works imagine that the buyer's type is drawn from some fixed publicly available distribution. Therefore learning $\mathcal{D}$ is not at issue. In contrast, we argue that access to an accurate prior is particularly problematic in these settings. After all, the seller cannot expect to reliably estimate $\mathcal{D}$ from data when the buyer is explicitly incentivized to hide its type (as illustrated in the discussion at the start of the chapter). This tension between learning and buyer truthfulness is in many ways central to our study.

Secondly, given a fixed prior, the most common solution concept in the BBPD literature is a perfect Bayes-Nash equilibrium, in which both the seller and buyer strategies are best

responses to each other. However, in the context of Internet advertising, a seller must first deploy an algorithm which automates the pricing strategy, and buyers subsequently react to the observed behavior of the pricing algorithm. Any modifications the seller wishes to make to the pricing algorithm will typically require changes to the end-user licensing agreement, which the seller will not want to do too frequently. Therefore, in this paper, we make a commitment assumption on the seller: the seller acts first, announcing its pricing strategy, after which the buyer plays a best response strategy. Such Stackleberg models of commitment Fudenberg and Tirole (1991) have sparked a great deal of recent interest due to their success in security games (see Conitzer and Sandholm (2006) and Korzhyk et al. (2011) for an overview), including practical deployment Pita et al. (2008); Jain et al. (2010).

Two settings that are distinct from what we consider in this dissertation, but where mechanism design and learning are combined, are the multi-armed bandit mechanism design problem Babaioff et al. (2009, 2010); Devanur and Kakade (2009) and the incentive compatible regression/classification problem Dekel et al. (2010); Meir et al. (2009). The former problem is motivated by sponsored search auctions, where the challenge is to elicit truthful values from multiple bidding advertisers while also efficiently estimating the click-through rate of the set of ads that are to be allocated. The latter problem involves learning a discriminative classifier or regression function in the batch setting with training examples that are labeled by selfish agents. The goal is then to minimize error with respect to the truthful labels.

Finally, Arora et al. proposed a notion of regret for online learning algorithms, called policy regret, that accounts for the possibility that the adversary may adapt to the learning algorithm's behavior Arora et al. (2012). This resembles the ability, in our setting, of a strategic buyer to adapt to the seller algorithm's behavior. However, even this stronger definition of regret is inadequate for our setting. This is because policy regret is equivalent to standard regret when the adversary is oblivious, and as we explained in the previous

section, there is an oblivious buyer strategy such that the seller's standard regret is small, but his regret with respect to the best fixed price against a truthful buyer is large.

5.2. Preliminaries and Model

We consider a posted-price model for a single buyer repeatedly purchasing items from a single seller. Associated with the buyer is a fixed distribution $\mathcal{D}$ over the interval $[0, 1]$, which is known only to the buyer. On each round t, the buyer receives a value $v_t \in \mathcal{V} \subseteq [0, 1]$ from the distribution $\mathcal{D}$. The seller, without observing this value, then posts a price $p_t \in \mathcal{P} \subseteq [0, 1]$. Finally, the buyer selects an allocation decision $a_t \in \{0, 1\}$. On each round t, the buyer receives an *instantaneous surplus* of $a_t(v_t - p_t)$, and the seller receives an *instantaneous revenue* of $a_t p_t$.

We will be primarily interested in designing the seller's *learning algorithm*, which we will denote $\mathcal{A}$. Let $v_{1:t}$ denote the sequence of values observed on the first t rounds, $(v_1, ..., v_t)$, defining $p_{1:t}$ and $a_{1:t}$ analogously. $\mathcal{A}$ is an algorithm that selects each price p_t as a (possibly randomized) function of $(p_{1:t-1}, a_{1:t-1})$. As is common in mechanism design, we assume that the seller announces his choice of algorithm $\mathcal{A}$ in advance. The buyer then selects her *allocation strategy* in response. The buyer's allocation strategy $\mathcal{B}$ generates allocation decisions a_t as a (possibly randomized) function of $(\mathcal{D}, v_{1:t}, p_{1:t}, a_{1:t-1})$.

Notice that a choice of $\mathcal{A}$, $\mathcal{B}$ and $\mathcal{D}$ fixes a distribution over the sequences $a_{1:T}$ and $p_{1:T}$. This in turn defines the seller's total expected revenue:

$$\text{SellerRevenue}(\mathcal{A}, \mathcal{B}, \mathcal{D}, T) = E\left[\sum_{t=1}^{T} a_t p_t \mid \mathcal{A}, \mathcal{B}, \mathcal{D}\right].$$

In the most general setting, we will consider a buyer whose surplus may be discounted through time. In fact, our lower bounds will demonstrate that a sufficiently decaying discount rate is necessary for a no-regret learning algorithm. We will imagine therefore that there exists a nonincreasing sequence $\{\gamma_t \in (0, 1]\}$ for the buyer. For a choice of T, we will define the effective "time-horizon" for the buyer as $T_\gamma = \sum_{t=1}^{T} \gamma_t$. The buyer's expected

total discounted surplus is given by:

$$\text{BuyerSurplus}(\mathcal{A}, \mathcal{B}, \mathcal{D}, T) = E\left[\sum_{t=1}^{T} \gamma_t a_t(v_t - p_t) \mid \mathcal{A}, \mathcal{B}, \mathcal{D}\right].$$

We assume that the seller is faced with a strategic buyer who adapts to the choice of $\mathcal{A}$. Thus, let $\mathcal{B}^*(\mathcal{A}, \mathcal{D})$ be a surplus-maximizing buyer for seller algorithm $\mathcal{A}$ and value distribution is $\mathcal{D}$. In other words, for all strategies $\mathcal{B}$ we have

$$\text{BuyerSurplus}(\mathcal{A}, \mathcal{B}^*(\mathcal{A}, \mathcal{D}), \mathcal{D}, T) \geq \text{BuyerSurplus}(\mathcal{A}, \mathcal{B}, \mathcal{D}, T).$$

We are now prepared to define the seller's regret. Let $p^* = \arg\max_{p \in \mathcal{P}} p \Pr_{\mathcal{D}}[v \geq p]$, the revenue-maximizing choice of price for a seller that *knows* the distribution $\mathcal{D}$, and simply posts a price of p^* on every round. Against such a pricing strategy, it is in the buyer's best interest to be *truthful*, accepting if and only if $v_t \geq p^*$, and the seller would receive a revenue of $Tp^* \Pr_{v \sim \mathcal{D}}[v \geq p^*]$. Informally, a no-regret algorithm is able to learn $\mathcal{D}$ from previous interactions with the buyer, and converge to selecting a price close to p^*. We therefore define regret as:

$$\text{Regret}(\mathcal{A}, \mathcal{D}, T) = Tp^* \Pr_{v \sim \mathcal{D}}[v \geq p^*] - \text{SellerRevenue}(\mathcal{A}, \mathcal{B}^*(\mathcal{A}, \mathcal{D}), \mathcal{D}, T).$$

Finally, we will be interested in algorithms that attain $o(T)$ regret (meaning the averaged regret goes to zero as $T \to \infty$) for the worst-case $\mathcal{D}$. In other words, we say $\mathcal{A}$ is *no-regret* if $\sup_{\mathcal{D}} \text{Regret}(\mathcal{A}, \mathcal{D}, T) = o(T)$. Note that this definition of worst-case regret only assumes that Nature's behavior (i.e., the value distribution) is worst-case; the buyer's behavior is always presumed to be surplus maximizing.

5.3. Fixed Value Setting

In this section we consider the case of a single unknown fixed buyer value, that is $\mathcal{V} = \{v\}$ for some $v \in (0, 1]$. We show that in this setting a very simple pricing algorithm with monotonically decreasing price offerings is able to achieve $O(T_\gamma \sqrt{T})$ when the buyer discount

is $\gamma_t = \gamma^{t-1}$. The full proofs for this section appear in Section 5.6.1.

> `Monotone` algorithm: Choose parameter $\beta \in (0,1)$, and initialize $a_0 = 1$ and $p_0 = 1$. In each round $t \geq 1$ let $p_t = \beta^{1-a_{t-1}} p_{t-1}$.

In the `Monotone` algorithm, the seller starts at the maximum price of 1, and decreases the price by a factor of β whenever the buyer rejects the price, and otherwise leaves it unchanged. Since `Monotone` is deterministic and the buyer's value v is fixed, the surplus-maximizing buyer algorithm $\mathcal{B}^*(\texttt{Monotone}, v)$ is characterized by a deterministic allocation sequence $a^*_{1:T} \in \{0,1\}^T$.[1]

The following lemma partially characterizes the optimal buyer allocation sequence.

Lemma 5.1. *The sequence $a^*_1, \ldots, a^*_T$ is monotonically nondecreasing.*

In other words, once a buyer decides to start accepting the offered price at a certain time step, she will keep accepting from that point on. The main idea behind the proof is to show that if there does exist some time step t' where $a^*_{t'} = 1$ and $a^*_{t'+1} = 0$, then swapping the values so that $a^*_{t'} = 0$ and $a^*_{t'+1} = 1$ (as well potentially swapping another pair of values) will result in a sequence with strictly better surplus, thereby contradicting the optimality of $a^*_{1:T}$. The full proof is shown in Section 5.6.1.

Now, to finish characterizing the optimal allocation sequence, we provide the following lemma, which describes time steps where the buyer has with certainty begun to accept the offered price.

Lemma 5.2. *Let $c_{\beta,\gamma} = 1 + (1-\beta)T_\gamma \quad$ and $\quad d_{\beta,\gamma} = \frac{\log\left(\frac{c_{\beta,\gamma}}{v}\right)}{\log(1/\beta)}$, then for any $t > d_{\beta,\gamma}$ we have $a^*_{t+1} = 1$.*

A detailed proof is presented in Section 5.6.1. These lemmas imply the following regret bound.

Theorem 5.1. $\text{Regret}(\texttt{Monotone}, v, T) \leq vT\left(1 - \frac{\beta}{c_{\beta,\gamma}}\right) + v\beta\left(\frac{d_{\beta,\gamma}}{c_{\beta,\gamma}} + \frac{1}{c_{\beta,\gamma}}\right)$.

[1] If there are multiple optimal sequences, the buyer can then choose to randomize over the set of sequences. In such a case, the worst case distribution (for the seller) is the one that always selects the revenue minimizing optimal sequence. In that case, let $a^*_{1:T}$ denote the revenue-minimizing buyer-optimal sequence.

Proof. By Lemmas 5.1 and 5.2 we receive no revenue until at most round $\lceil d_{\beta,\gamma} \rceil + 1$, and from that round onwards we receive at least revenue $\beta^{\lceil d_{\beta,\gamma} \rceil}$ per round. Thus

$$\text{Regret}(\texttt{Monotone}, v, T) = vT - \sum_{t=\lceil d_{\beta,\gamma} \rceil + 1}^{T} \beta^{\lceil d_{\beta,\gamma} \rceil} \leq vT - (T - d_{\beta,\gamma} - 1)\beta^{d_{\beta,\gamma}+1}$$

Noting that $\beta^{d_{\beta,\gamma}} = \frac{v}{c_{\beta,\gamma}}$ and rearranging proves the theorem. □

Tuning the learning parameter simplifies the bound further and provides a $O(T_\gamma \sqrt{T})$ regret bound. Note that this tuning parameter does not assume knowledge of the buyer's discount parameter γ.

Corollary 5.1. *If* $\beta = \frac{\sqrt{T}}{1+\sqrt{T}}$ *then* $\text{Regret}(\texttt{Monotone}, v, T) \leq \sqrt{T}\left(4vT_\gamma + 2v \log\left(\frac{1}{v}\right)\right) + v\,.$

The computation used to derive this corollary are found in Section 5.6.1. This corollary shows that it is indeed possible to achieve no-regret against a strategic buyer with a unknown fixed value as long as $T_\gamma = o(\sqrt{T})$. That is, the effective buyer horizon must be more than a constant factor smaller than the square-root of the game's finite horizon.

5.4. Upper Bound on Regret of `Phased`

We next give a seller algorithm that attains no-regret when the set of prices $\mathcal{P}$ is finite, the buyer's discount is $\gamma_t = \gamma^{t-1}$, and the buyer's value v_t for each round is drawn from a fixed distribution $\mathcal{D}$ that satisfies a certain continuity assumption, detailed below.

> `Phased` algorithm: Choose parameter $\alpha \in (0, 1)$. Define $T_i \equiv 2^i$ and $S_i \equiv \min\left(\frac{T_i}{|\mathcal{P}|}, T_i^\alpha\right)$. For each phase $i = 1, 2, 3, \ldots$ of length T_i rounds:
>
> Offer each price $p \in \mathcal{P}$ for S_i rounds, in some fixed order; these are the *explore* rounds. Let $A_{p,i} =$ Number of explore rounds in phase i where price p was offered and the buyer accepted. For the remaining $T_i - |\mathcal{P}|S_i$ rounds of phase i, offer price $\tilde{p}_i = \arg\max_{p \in \mathcal{P}} p\frac{A_{p,i}}{S_i}$ in each round; these are the *exploit* rounds.

The `Phased` algorithm proceeds across a number of phases. Each phase consists of explore rounds followed by exploit rounds. During explore rounds, the algorithm selects each price in some fixed order. During exploit rounds, the algorithm repeatedly selects the price that realized the greatest revenue during the immediately preceding explore rounds.

First notice that a strategic buyer has no incentive to lie during exploit rounds (i.e. it will accept any price $p_t < v_t$ and reject any price $p_t > v_t$), since its decisions there do not affect any of its future prices. Thus, the exploit rounds are the time at which the seller can exploit what it has learned from the buyer during exploration. Alternatively, if the buyer has successfully manipulated the seller into offering a low price, we can view the buyer as "exploiting" the seller.

During explore rounds, on the other hand, the strategic buyer can benefit by telling lies which will cause it to witness better prices during the corresponding exploit rounds. However, the value of these lies to the buyer will depend on the fraction of the phase consisting of explore rounds. Taken to the extreme, if the entire phase consists of explore rounds, the buyer is not interested in lying. In general, the more explore rounds, the more revenue has to be sacrificed by a buyer that is lying during the explore rounds. For the myopic buyer, the loss of enough immediate revenue at some point ceases to justify her potential gains in the future exploit rounds.

Thus, while traditional algorithms like UCB balance exploration and exploitation to ensure confidence in the observed payoffs of sampled arms, our `Phased` algorithm explores for two purposes: to ensure accurate estimates, and to dampen the buyer's incentive to mislead the seller. The seller's balancing act is to explore for long enough to learn the buyer's value distribution, but leave enough exploit rounds to benefit from the knowledge.

Continuity of the value distribution The preceding argument required that the distribution $\mathcal{D}$ does not exhibit a certain pathology. There cannot be two prices p, p' that are very close but $p \Pr_{v\sim\mathcal{D}}[v \geq p]$ and $p' \Pr_{v\sim\mathcal{D}}[v \geq p']$ are very different. Otherwise, the buyer

is largely indifferent to being offered prices p or p', but distinguishing between the two prices is essential for the seller during exploit rounds. Thus, we assume that the value distribution $\mathcal{D}$ is *K-Lipschitz*, which eliminates this problem: Defining $F(p) \equiv \Pr_{v\sim\mathcal{D}}[v \geq p]$, we assume there exists $K > 0$ such that $|F(p) - F(p')| \leq K|p - p'|$ for all $p, p' \in [0, 1]$. This assumption is quite mild, as our `Phased` algorithm does not need to know K, and the dependence of the regret rate on K will be logarithmic.

Theorem 5.2. *Assume $F(p) \equiv \Pr_{v\sim\mathcal{D}}[v \geq p]$ is K-Lipschitz. Let $\Delta = \min_{p\in\mathcal{P}\setminus\{p^*\}} p^*F(p^*) - pF(p)$, where $p^* = \arg\max_{p\in\mathcal{P}} pF(p)$. For any parameter $\alpha \in (0, 1)$ of the* `Phased` *algorithm there exist constants c_1, c_2, c_3, c_4 such that*

$$\begin{aligned}\text{Regret}(\texttt{Phased}, \mathcal{D}, T) \leq\ & c_1|\mathcal{P}|T^\alpha + c_2\frac{|\mathcal{P}|^2}{\Delta^{2/\alpha}}(\log T)^{1/\alpha} \\ & + c_3\frac{|\mathcal{P}|^2}{\Delta^{1/\alpha}}T_\gamma^{1/\alpha}(\log T + \log(K/\Delta))^{1/\alpha} + c_4|\mathcal{P}| \\ =\ & \tilde{O}(T^\alpha + T_\gamma^{1/\alpha}).\end{aligned}$$

The complete proof of Theorem 5.2 is rather technical, and is provided in Appendix 5.6.2.

To gain further intuition about the upper bounds proved in this section and the previous section, it helps to parametrize the buyer's horizon T_γ as a function of T, e.g. $T_\gamma = T^c$ for $0 \leq c \leq 1$. Writing it in this fashion, we see that the `Monotone` algorithm has regret at most $O(T^{c+\frac{1}{2}})$, and the `Phased` algorithm has regret at most $\tilde{O}(T^{\sqrt{c}})$ if we choose $\alpha = \sqrt{c}$. The lower bound proved in the next section states that, in the worst case, any seller algorithm will incur a regret of at least $\Omega(T^c)$.

5.5. Lower Bound

In this section we state the main lower bound, which establishes a connection between the regret of any seller algorithm and the buyer's discounting. Specifically, we prove that the regret of any seller algorithm is $\Omega(T_\gamma)$. Note that when $T = T_\gamma$ — i.e., the buyer does not discount her future surplus — our lower bound proves that no-regret seller algorithms do

not exist, and thus it is *impossible for the seller to take advantage of learned information.* For example, consider the seller algorithm that uniformly selects prices p_t from $[0, 1]$. The optimal buyer algorithm is truthful, accepting if $p_t < v_t$, as the seller algorithm is non-adaptive, and the buyer does not gain any advantage by being more strategic. In such a scenario the seller would quickly learn a good estimate of the value distribution $\mathcal{D}$. What is surprising is that a seller cannot *use* this information if the buyer does not discount her future surplus. If the seller attempts to leverage information learned through interactions with the buyer, the buyer can react accordingly to negate this advantage.

The lower bound further relates regret in the repeated setting to regret in a particular single-shot game between the buyer and the seller. This demonstrates that, against a non-discounted buyer, the seller is no better off in the repeated setting than he would be by repeatedly implementing such a single-shot mechanism (ignoring previous interactions with the buyer). In the following section we describe the simple single-shot game.

5.5.1. Single-Shot Auction

We call the following game the *single-shot auction.* A seller selects a family of distributions $\mathcal{S}$ indexed by $b \in [0, 1]$, where each $\mathcal{S}_b$ is a distribution on $[0, 1] \times \{0, 1\}$. The family $\mathcal{S}$ is revealed to a buyer with unknown value $v \in [0, 1]$, who then must select a bid $b \in [0, 1]$, and then $(p, a) \sim \mathcal{S}_b$ is drawn from the corresponding distribution.

As usual, the buyer gets a surplus of $a(v - p)$, while the seller enjoys a revenue of ap. We restrict the set of seller strategies to distributions that are *incentive compatible* and *rational.* $\mathcal{S}$ is incentive compatible if for all $b, v \in [0, 1]$, $E_{(p,a)\sim\mathcal{S}_b}[a(v-p)] \leq E_{(p,a)\sim\mathcal{S}_v}[a(v-p)]$. It is *rational* if for all v, $E_{(p,a)\sim\mathcal{S}_v}[a(v-p)] \geq 0$ (i.e. any buyer maximizing expected surplus is actually incentivised to play the game). Incentive compatible and rational strategies exist: drawing p from a fixed distribution (i.e. all $\mathcal{S}_b$ are the same), and letting $a = \mathbf{1}\{b \geq p\}$ suffices.[2]

[2]This subclass of auctions is even *ex post* rational.

We define the regret in the single-shot setting of any incentive-compatible and rational strategy $\mathcal{S}$ with respect to value v as

$$\text{SSRegret}(\mathcal{S}, v) = v - E_{(p,a)\sim\mathcal{S}_v}[ap].$$

The following loose lower bound on $\text{SSRegret}(\mathcal{S}, v)$ is straightforward, and establishes that a seller's revenue cannot be a constant fraction of the buyer's value for all v. The full proof is provided in the appendix (Section 5.6.3).

Lemma 5.3. *For any incentive compatible and rational strategy $\mathcal{S}$ there exists $v \in [0, 1]$ such that* $\text{SSRegret}(\mathcal{S}, v) \geq \frac{1}{12}$.

5.5.2. Repeated Auction

Returning to the repeated setting, our main lower bound will make use of the following technical lemma, the full proof of which is provided in the appendix (Section 5.6.3). Informally, the Lemma states that the surplus enjoyed by an optimal buyer algorithm would only increase if this surplus were viewed without discounting.

Lemma 5.4. *Let the buyer's discount sequence $\{\gamma_t\}$ be positive and nonincreasing. For any seller algorithm $\mathcal{A}$, value distribution $\mathcal{D}$, and surplus-maximizing buyer algorithm $\mathcal{B}^*(\mathcal{A}, \mathcal{D})$,* $E\left[\sum_{t=1}^{T} \gamma_t a_t(v_t - p_t)\right] \leq E\left[\sum_{t=1}^{T} a_t(v_t - p_t)\right]$

Notice if $a_t(v_t - p_t) \geq 0$ for all t, then the Lemma 5.4 is trivial. This would occur if the buyer only ever accepts prices less than its value ($a_t = 1$ only if $p_t \leq v_t$). However, Lemma 5.4 is interesting in that it holds for *any* seller algorithm $\mathcal{A}$. It's easy to imagine a seller algorithm that incentivizes the buyer to sometimes accept a price $p_t > v_t$ with the promise that this will generate better prices in the future (e.g. setting $p_{t'} = 1$ and offering $p_t = 0$ for all $t > t'$ only if $a_{t'} = 1$ and otherwise setting $p_t = 1$ for all $t > t'$).

Lemmas 5.3 and 5.4 let us prove our main lower bound.

Theorem 5.3. *Fix a positive, nonincreasing, discount sequence $\{\gamma_t\}$. Let $\mathcal{A}$ be any seller*

algorithm for the repeated setting. There exists a buyer value distribution $\mathcal{D}$ such that $\text{Regret}(\mathcal{A}, \mathcal{D}, T) \geq \frac{1}{12} T_\gamma$. *In particular, if $T_\gamma = \Omega(T)$, no-regret is impossible.*

Proof. Let $\{a_{b,t}, p_{b,t}\}$ be the sequence of prices and allocations generated by playing $\mathcal{B}^*(\mathcal{A}, b)$ against $\mathcal{A}$. For each $b \in [0,1]$ and $(p,a) \in [0,1) \times \{0,1\}$, let $\mu_b(p,a) = \frac{1}{T_\gamma} \sum_{t=1}^{T} \gamma_t \mathbf{1}\{a_{b,t} = a\}\mathbf{1}\{p_{b,t} = p\}$. Notice that $\mu_b(p,a) > 0$ for countably many (p,a) and let $\Omega_b = \{(p,a) \in [0,1] \times \{0,1\} : \mu_b(p,a) > 0\}$. We think of μ_b as being a distribution. It's in fact a random measure since the $\{a_{b,t}, p_{b,t}\}$ are themselves random. One could imagine generating μ_b by playing $\mathcal{B}^*(\mathcal{A}, b)$ against $\mathcal{A}$ and observing the sequence $\{a_{b,t}, p_{b,t}\}$. Every time we observe a price $p_{b,t} = p$ and allocation $a_{b,t} = a$, we assign $\frac{1}{T_\gamma}\gamma_t$ additional mass to (p,a) in μ_b. This is impossible in practice, but the random measure μ_b has a well-defined distribution.

Now consider the following strategy $\mathcal{S}$ for the single-shot setting. $\mathcal{S}_b$ is induced by drawing a μ_b, then drawing $(p,a) \sim \mu_b$. Note that for any $b \in [0,1]$ and any measurable function f

$$E_{(p,a)\sim\mathcal{S}_b}[f(a,p)] = E_{\mu_b \sim S_b}\big[E_{(p,a)\sim\mu_b}[f(a,b) \mid \mu_b]\big] = \tfrac{1}{T_\gamma} E\Big[\textstyle\sum_{t=1}^{T} \gamma_t f(a_{b,t}, p_{b,t})\Big].$$

Thus the strategy $\mathcal{S}$ is incentive compatible, since for any $b, v \in [0,1]$

$$E_{(p,a)\sim\mathcal{S}_b}[a(v-p)] = \frac{1}{T_\gamma} E\left[\sum_{t=1}^{T} \gamma_t a_{b,t}(v - p_{b,t})\right] = \frac{1}{T_\gamma}\text{BuyerSurplus}(\mathcal{A}, \mathcal{B}^*(\mathcal{A}, b), v, T)$$
$$\leq \frac{1}{T_\gamma}\text{BuyerSurplus}(\mathcal{A}, \mathcal{B}^*(\mathcal{A}, v), v, T) = \frac{1}{T_\gamma} E\left[\sum_{t=1}^{T} \gamma_t a_{v,t}(v - p_{v,t})\right] = E_{(p,a)\sim\mathcal{S}_v}[a(v-p)]$$

where the inequality follows from the fact that $\mathcal{B}^*(\mathcal{A}, v)$ is a surplus-maximizing algorithm for a buyer whose value is v. The strategy $\mathcal{S}$ is also rational, since for any $v \in [0,1]$

$$E_{(p,a)\sim\mathcal{S}_v}[a(v-p)] = \frac{1}{T_\gamma} E\left[\sum_{t=1}^{T} \gamma_t a_{v,t}(v - p_{v,t})\right] = \frac{1}{T_\gamma}\text{BuyerSurplus}(\mathcal{A}, \mathcal{B}^*(\mathcal{A}, v), v, T) \geq 0$$

where the inequality follows from the fact that a surplus-maximizing buyer algorithm cannot earn negative surplus, as a buyer can always reject every price and earn zero surplus.

Let $r_t = 1-\gamma_t$ and $T_r = \sum_{t=1}^{T} r_t$. Note that $r_t \geq 0$. We have the following for any $v \in [0,1]$:

$$
\begin{aligned}
T_\gamma \text{SSRegret}(\mathcal{S}, v) &= T_\gamma \left(v - E_{(p,a)\sim\mathcal{S}_v}[ap]\right) = T_\gamma \left(v - \frac{1}{T_\gamma} E\left[\sum_{t=1}^{T} \gamma_t a_{v,t} p_{v,t}\right]\right) \\
&= T_\gamma v - E\left[\sum_{t=1}^{T} \gamma_t a_{v,t} p_{v,t}\right] = (T - T_r)v - E\left[\sum_{t=1}^{T} (1-r_t) a_{v,t} p_{v,t}\right] \\
&= Tv - E\left[\sum_{t=1}^{T} a_{v,t} p_{v,t}\right] + E\left[\sum_{t=1}^{T} r_t a_{v,t} p_{v,t}\right] - T_r v \\
&= \text{Regret}(\mathcal{A}, v, T) + E\left[\sum_{t=1}^{T} r_t a_{v,t} p_{v,t}\right] - T_r v = \text{Regret}(\mathcal{A}, v, T) + E\left[\sum_{t=1}^{T} r_t (a_{v,t} p_{v,t} - v)\right]
\end{aligned}
$$

A closer look at the quantity $E\left[\sum_{t=1}^{T} r_t (a_{v,t} p_{v,t} - v)\right]$, tells us that: $E\left[\sum_{t=1}^{T} r_t (a_{v,t} p_{v,t} - v)\right] \leq E\left[\sum_{t=1}^{T} r_t a_{v,t} (p_{v,t} - v)\right] = -E\left[\sum_{t=1}^{T} (1-\gamma_t) a_{v,t} (v - p_{v,t})\right] \leq 0$, where the last inequality follows from Lemma 5.4. Therefore $T_\gamma \text{SSRegret}(\mathcal{S}, v) \leq \text{Regret}(\mathcal{A}, v, T)$ and taking $\mathcal{D}$ to be the point-mass on the value $v \in [0,1]$ which realizes Lemma 5.3 proves the statement of the theorem. □

5.6. Detailed Proofs

5.6.1. Upper Bound on the Regret of `Monotone`

Restatement of Lemma 5.1: *The sequence $a_1^*, \ldots, a_T^*$ is monotonically nondecreasing.*

Proof. For any sequence $\mathbf{a} \in \{0,1\}^T$ let $\text{last}(\mathbf{a})$ be the last round t where $a_t = 1$ and $a_{t+1} = 0$, or $\text{last}(\mathbf{a}) = 0$ if there is no such round. Let $\mathbf{a}^* = a_1^*, \ldots, a_T^*$, and assume for contradiction that $\text{last}(\mathbf{a}^*) > 0$. Further, assume without loss of generality that $\text{last}(\mathbf{a}^*) \geq \text{last}(\tilde{\mathbf{a}}^*)$ for every optimal sequence $\tilde{\mathbf{a}}^*$. Let $\ell = \text{last}(\mathbf{a}^*)$.

Suppose that $a_t^* = 0$ for all $t \geq \ell + 1$. If $v - p_\ell \geq 0$ then, since $p_{\ell+1} = p_\ell$, letting $a_{\ell+1}^* = 1$ does not decrease the buyer's total surplus and increases $\text{last}(\mathbf{a}^*)$, violating the assumption that $\text{last}(\mathbf{a}^*) \geq \text{last}(\tilde{\mathbf{a}}^*)$ for every optimal sequence $\tilde{\mathbf{a}}^*$. On the other hand, if $v - p_\ell < 0$ then letting $a_\ell^* = 0$ increases the buyer's total surplus, contradicting the optimality of $\mathbf{a}^*$.

Otherwise choose the smallest $k \geq 1$ such that $a^*_{\ell+k} = 0$, and $a^*_{\ell+k+1} = 1$. Note that $p_{\ell+k+1} = \beta^k p_\ell$ and $p_{\ell+k} = \beta^{\ell-1} p_\ell$. Swapping the values of a^*_ℓ and $a^*_{\ell+1}$ does not affect the buyer's surplus in rounds other than ℓ and $\ell+1$, and must not increase the buyer's total surplus, which implies $\gamma^{\ell-1}(v - p_\ell) \geq \gamma^\ell(v - \beta p_\ell)$. Likewise, swapping the values of $a^*_{\ell+k}$ and $a^*_{\ell+k+1}$ does not affect the buyer's surplus in rounds other than $\ell+k$ and $\ell+k+1$, and increases last($\mathbf{a}^*$), so it must decrease the buyer's total surplus, which implies $\gamma^{\ell+k}(v - p_{\ell+k+1}) > \gamma^{\ell+k-1}(v - p_{\ell+k})$.

Cancelling γ's in each inequality, and substituting for $p_{\ell+k}$ and $p_{\ell+k+1}$ gives the following inequalities:

$$v - p_\ell \geq \gamma v - \gamma\beta p_\ell \quad \text{ and } \gamma v - \gamma\beta^k p_\ell > v - \beta^{k-1} p_\ell$$

Adding the two inequalities and rearranging gives us:

$$\beta^{k-1} p_\ell + \gamma p_\ell(\beta - \beta^k) > p_\ell$$

Dividing through by p_ℓ gives us:

$$\beta^{k-1} + \gamma(\beta - \beta^k) > 1 \tag{5.1}$$

Let $g(\beta) = \beta^{k-1} + \beta - \beta^k$. Since $\beta - \beta^k$ is non-negative and $\gamma \leq 1$, $g(\beta)$ is an upper bound on the left hand side of equation 5.1. Giving:

$$\beta^{k-1} + \gamma(\beta - \beta^k) \leq g(\beta) \tag{5.2}$$

However, $\frac{dg}{d\beta} = (k-1)\beta^{k-2} + 1 - k\beta^{k-1} = (1-\beta^{k-2}) + k(\beta^{k-2} - \beta^{k-1})$, which is non-negative for any $\beta < 1$. To see why, note that both terms in the last expression are non-negative when $k > 1$ and the entire expression is 0 when $k = 1$.

Therefore, $g(\cdot)$ is a non-decreasing function and for any $\beta < 1$, $g(\beta) \leq g(1) = 1$. This fact

combined with Eq. (5.1) and Eq. (5.2) imply a contradiction. □

Restatement of Lemma 5.2 *Let* $c_{\beta,\gamma} = 1 + (1-\beta)T_\gamma \quad$ *and* $\quad d_{\beta,\gamma} = \frac{\log\left(\frac{c_{\beta,\gamma}}{v}\right)}{\log(1/\beta)}$*, then for any* $t > d_{\beta,\gamma}$ *we have* $a^*_{t+1} = 1$.

Proof. Rearranging the inequality $t > d_{\beta,\gamma}$ yields $\beta^t\left(1 + (1-\beta)T_\gamma\right) < v$. Subtracting β^{t+1} from both sides, multiplying both sides by γ^t, and applying the inequality $\sum_{t'=1}^{T-t}\gamma^{t'-1} \leq \sum_{t'=1}^{T}\gamma^{t'-1} = T_\gamma$ gives us

$$\begin{aligned} & \gamma^t\left(\beta^t\left(1 + (1-\beta)\sum_{t'=1}^{T-t-1}\gamma^{t'-1}\right) - \beta^{t+1}\right) < \gamma^t(v - \beta^{t+1}) \\ \Leftrightarrow \quad & \beta^t(1-\beta)\sum_{t'=t+1}^{T}\gamma^{t'-1} < \gamma^t(v-\beta^{t+1}) \end{aligned}$$

Now substitute $\beta^t(1-\beta) = (v-\beta^{t+1}) - (v-\beta^t)$ and gather terms. We have

$$\sum_{t'=t+2}^{T}\gamma^{t'-1}(v-\beta^{t+1}) < \sum_{t'=t+1}^{T}\gamma^{t'-1}(v-\beta^t)\,. \tag{5.3}$$

Note that $\sum_{t'=t+1}^{T}\gamma^{t'-1}(v-\beta^t)$ is the surplus of a monotonic buyer that starts accepting (and thus continues to accept) the price offered at time $t+1$. The inequality above, which holds for arbitrary $t > d_{\beta,\gamma}$, states that the surplus that is gained from starting to accept at round $t+1$ is greater than the surplus gained from starting to accept at round $t+2$. Thus, it must be the case $a^*_{t+1} = 1$. □

Restatement of Corollary 5.1 *If* $\beta = \frac{\sqrt{T}}{1+\sqrt{T}}$ *then*

$$\text{Regret}(\texttt{Monotone}, v, T) \leq \sqrt{T}\left(4vT_\gamma + 2v\log\left(\frac{1}{v}\right)\right) + v\,.$$

Before showing the proof to Corollary 5.1, we prove the following technical lemma.

Lemma 5.5. *$x \geq \log(1+x)$ if $x \geq 0$ and $x \leq 2\log(1+x)$ if $0 \leq x \leq 1$.*

Proof. By Taylor's theorem $e^x = \sum_{i=0}^{\infty} \frac{x^i}{i!}$. Therefore $e^x \geq 1 + x$ if $x \geq 0$, and so $x \geq \log(1+x)$ if $x \geq 0$. Now let $a_n = \sum_{i=1}^{n} (-1)^{i+1} \frac{x^i}{i}$ and observe that for any positive even integer n

$$\begin{aligned} 2a_n &= 2x - x^2 + 2\sum_{i=3}^{n} (-1)^{i+1} \frac{x^i}{i} \\ &= x + \left(x - x^2\right) + 2 \sum_{i=3,5,7,\ldots}^{n} x^i \left(\frac{1}{i} - \frac{x}{i+1}\right) \\ &\geq x \end{aligned}$$

where the inequality follows because $x - x^2 \geq 0$ if $0 \leq x \leq 1$ and $\frac{1}{i} - \frac{x}{i+1} \geq 0$ if $x \leq 1$ and $i \geq 1$. Since $\lim_{n\to\infty} a_n = \log(1+x)$ (by Taylor's theorem) and $\lim_{n\to\infty} a_n = \lim_{n\to\infty, n \text{ even}} a_n$ (because all subsequences of a convergent sequence have the same limit), we have shown $2\log(1+x) \geq x$ for $0 \leq x \leq 1$. □

Now, the proof of Corollary 5.1.

Proof of Corollary 5.1. From the expression for β we have

$$c_{\beta,\gamma} = 1 + \left(1 - \frac{\sqrt{T}}{1+\sqrt{T}}\right) T_\gamma = 1 + \frac{1}{\left(1+\sqrt{T}\right)} T_\gamma = \frac{1 + \sqrt{T} + T_\gamma}{\left(1+\sqrt{T}\right)} \tag{5.4}$$

which implies

$$1 - \frac{\beta}{c_{\beta,\gamma}} = 1 - \frac{\sqrt{T}}{1 + \sqrt{T} + T_\gamma} = \frac{1 + T_\gamma}{1 + \sqrt{T} + T_\gamma}.$$

We also have

$$d_{\beta,\gamma} = \frac{\log\left(\left(1 + \frac{T_\gamma}{(1+\sqrt{T})}\right)\frac{1}{v}\right)}{\log\left(\frac{1+\sqrt{T}}{\sqrt{T}}\right)} = \frac{\log\left(1 + \frac{T_\gamma}{(1+\sqrt{T})}\right) + \log\left(\frac{1}{v}\right)}{\log\left(1 + \frac{1}{\sqrt{T}}\right)}.$$

By Lemma 5.5 we know that $x \geq \log(1+x)$ if $x \geq 0$ and $x \leq 2\log(1+x)$ if $0 \leq x \leq 1$.

Since $T \geq 1$ we have $\frac{T_\gamma}{\left(1+\sqrt{T}\right)} \geq 0$ and $0 \leq \frac{1}{\sqrt{T}} \leq 1$ and therefore

$$d_{\beta,\gamma} \leq \frac{2T_\gamma\sqrt{T}}{\left(1+\sqrt{T}\right)} + 2\sqrt{T}\log\left(\frac{1}{v}\right) \leq 2T_\gamma + 2\sqrt{T}\log\left(\frac{1}{v}\right). \tag{5.5}$$

From the expression for $c_{\beta,\gamma}$ in Eq. (5.4) we have $\frac{1}{c_{\beta,\gamma}} \leq 1$. Therefore

$$\frac{d_{\beta,\gamma}}{c_{\beta,\gamma}} \leq 2T_\gamma + 2\sqrt{T}\log\left(\frac{1}{v}\right).$$

Now plug the bounds on $1 - \frac{\beta}{c_{\beta,\gamma}}$, $\frac{d_{\beta,\gamma}}{c_{\beta,\gamma}}$ and $\frac{1}{c_{\beta,\gamma}}$ from above into the upper bound from Theorem 5.1. Noting that $\beta \leq 1$ gives us

$$\begin{aligned}\text{Regret}(\texttt{Monotone}, v, T) &\leq vT\left(\frac{1+T_\gamma}{1+\sqrt{T}+T_\gamma}\right) + v\beta\left(2T_\gamma + 2\sqrt{T}\log\left(\frac{1}{v}\right)+1\right)\\ &\leq \sqrt{T}\left(4vT_\gamma + 2v\log\left(\frac{1}{v}\right)\right) + v\,. \end{aligned}$$ □

5.6.2. Upper Bound on Regret of `Phased`

Let λ be a fixed positive constant, whose exact value will be specified later. Define $V^+_{p,i}$ to be the number of explore rounds in phase i where price p was offered and the buyer's value in the round was at least $p+\lambda$. Let $\hat{r}^+_{p,i} = p\frac{V^+_{p,i}}{S_i}$, and note that $E[\hat{r}^+_{p,i}] = pF(p+\lambda)$. Similarly, define $V^-_{p,i}$ to be the number of explore rounds in phase i where price p was offered and the buyer's value in the round was at least $p-\lambda$. Let $\hat{r}^-_{p,i} = p\frac{V^-_{p,i}}{S_i}$, and note that $E[\hat{r}^-_{p,i}] = pF(p-\lambda)$. Also, let $A_{p,i}$ be the number of explore rounds in phase i where price p was offered and was accepted by the buyer. Then, we let $\tilde{r}_{p,i} = p\frac{A_{p,i}}{S_i}$ denote the *observed* revenue of price p in explore rounds in phase i.

In the `Phased` algorithm, the price $\tilde{p}_i$ that maximizes $\tilde{r}_{p,i}$ is offered in every exploit round of phase i. So our strategy for proving Theorem 5.2 will be to show that $p^* = \arg\max_p \tilde{r}_{p,i}$ with high probability for all sufficiently large i. There are essentially only two ways this can fail to happen: Either the realized buyer values differ greatly from their expectations, or the

buyer is untruthful about her realized values. The first case is unlikely, and the latter case is costly to the buyer, provided the number of explore rounds in the phase is sufficiently large. We now quantify "sufficiently large". Let i^* be the smallest nonnegative integer such that $S_i \geq D_T$ for all $i \geq i^*$, where

$$D_T = \max\left(\frac{16}{\Delta^2}\log T,\ \frac{8}{\Delta}C_{\frac{1}{T}}\right)$$

and $C_\delta = \log(1 + (1-\gamma)T_\gamma/(\delta\lambda))\log(1/\gamma)^{-1}$. Note that i^* is well-defined because S_i is increasing in i. The next lemma uses a standard concentration inequality to bound the probability that certain random variables are close to their expectations.

Lemma 5.6. *Fix price $p \in P$ and phase $i \geq i^*$. With probability $1 - 2T^{-1}$*

$$\hat{r}^-_{p,i} \leq pF(p-\lambda) + \frac{\Delta}{4} \quad \textit{and} \quad \hat{r}^+_{p^*,i} \geq p^*F(p^*+\lambda) - \frac{\Delta}{4}.$$

Proof. Note that $\hat{r}^-_{p,i}$ is an average of S_i independent random variables, since the variables p_t are chosen deterministically during the explore phase and each v_t is always drawn independently. Also note that $E[\hat{r}^-_{p,i}] = pF(p-\lambda)$. Since $i \geq i^*$ we have

$$S_i \geq \frac{16}{\Delta^2}\log T = \frac{1}{(\Delta/4)^2}\log T.$$

Thus by Hoeffding's inequality $\Pr\left[\hat{r}^-_{p,i} \leq pF(p-\lambda) + \frac{\Delta}{4}\right] \geq 1 - T^{-1}$. Similarly $\hat{r}^+_{p^*,i}$ is an average of S_i independent random variables and $E[\hat{r}^+_{p^*,i}] = p^*F(p^*+\lambda)$, and thus $\Pr\left[\hat{r}^+_{p^*,i} \geq p^*F(p^*+\lambda) - \frac{\Delta}{4}\right] \geq 1 - T^{-1}$. The lemma follows from the union bound. □

Let $\mathcal{L}_{p,i}$ be the set of explore rounds in phase i where the seller offered price p and the buyer λ-*lied*, i.e., a round t where either the buyer accepted price p and her value $v_t \leq p - \lambda$, or rejected price p and her value $v_t > p + \lambda$. Let $L_{p,i} = |\mathcal{L}_{p,i}|$. The next lemma shows that, for any phase i where the event from the previous lemma occurs, if the observed revenue of the optimal price p^* is less than the observed revenue of another price then the buyer must have told many λ-lies during phase i.

Lemma 5.7. *Fix price $p \in P$ and phase i. If $\tilde{r}_{p^*,i} < \tilde{r}_{p,i}$ and the event from Lemma 5.6 occurs then $L_{p,i} \geq \left(\frac{\Delta - 4K\lambda}{4p}\right) S_i$ or $L_{p^*,i} \geq \left(\frac{\Delta - 4K\lambda}{4p^*}\right) S_i$.*

Proof. Assume for contradiction that $L_{p,i} < \left(\frac{\Delta - 4K\lambda}{4p}\right) S_i$ and $L_{p^*,i} < \left(\frac{\Delta - 4K\lambda}{4p^*}\right) S_i$. For any price p' note that $A_{p',i} - V^-_{p',i} \leq L_{p',i}$ and $V^+_{p',i} - A_{p',i} \leq L_{p',i}$, since $A_{p',i}$ counts the number of times the buyer accepted price p' in phase i. Combining these bounds and applying the definitions of $\tilde{r}_{p,i}, \tilde{r}_{p^*,i}, \hat{r}^-_{p,i}$ and $\hat{r}^+_{p^*,i}$ proves

$$\tilde{r}_{p,i} - \hat{r}^-_{p,i} = \frac{p}{S_i}\Big(A_{p,i} - V^-_{p,i}\Big) < \frac{p}{S_i}\Big(\frac{\Delta - 4K\lambda}{4p}\Big)S_i = \frac{\Delta}{4} - K\lambda, \tag{5.6}$$

$$\hat{r}^+_{p^*,i} - \tilde{r}_{p^*,i} = \frac{p^*}{S_i}\Big(V^+_{p^*,i} - A_{p^*,i}\Big) < \frac{p^*}{S_i}\Big(\frac{\Delta - 4K\lambda}{4p^*}\Big)S_i = \frac{\Delta}{4} - K\lambda. \tag{5.7}$$

Now observe

$$\begin{aligned}
\tilde{r}_{p,i} &< \hat{r}^-_{p,i} + \frac{\Delta}{4} - K\lambda && \text{Eq. (5.6)} \\
&\leq pF(p-\lambda) + \frac{\Delta}{2} - K\lambda && \text{Lemma 5.6} \\
&\leq p(F(p) + K\lambda) + \frac{\Delta}{2} - K\lambda && K\text{-Lipschitz continuity} \\
&\leq pF(p) + \frac{\Delta}{2} \\
&\leq p^*F(p^*) - \frac{\Delta}{2} && \text{Definition of } \Delta \\
&\leq p^*(F(p^*+\lambda) + K\lambda) - \frac{\Delta}{2} && K\text{-Lipschitz continuity} \\
&\leq p^*F(p^*+\lambda) - \frac{\Delta}{2} + K\lambda \\
&\leq \hat{r}^+_{p^*,i} - \frac{\Delta}{4} + K\lambda && \text{Lemma 5.6} \\
&< \tilde{r}_{p^*,i} && \text{Eq. (5.7)}
\end{aligned}$$

which contradicts $\tilde{r}_{p^*,i} < \tilde{r}_{p,i}$. □

Next we show that the number of λ-lies told by a surplus-maximizing buyer in any phase

is bounded with high probability. This is the main technical lemma.

Lemma 5.8. *Fix price $p \in \mathcal{P}$, phase i, and suppose the buyer uses a surplus-maximizing algorithm $\mathcal{B}^*(\texttt{Phased}, \mathcal{D})$. For all $\delta > 0$ we have $\Pr[L_{p,i} \geq C_\delta] \leq \delta$.*

Proof. Let $\mathcal{B}^i$ be a buyer algorithm that acts according to $\mathcal{B}^*(\texttt{Phased}, \mathcal{D})$ during the first $i-1$ phases, and from phase i onwards acts truthfully in every round, i.e., $a_t = \mathbf{1}\{v_t \geq p_t\}$ for all rounds t in phases $i, i+1, \ldots, \lceil \log_2 T \rceil$. Assume $\Pr[L_{p,i} \geq C_\delta] > \delta$. We will show that this implies

$$\text{BuyerSurplus}(\texttt{Phased}, \mathcal{B}^*(\texttt{Phased}, \mathcal{D}), \mathcal{D}, T) < \text{BuyerSurplus}(\texttt{Phased}, \mathcal{B}^i, \mathcal{D}, T),$$

a contradiction.

Let $p_1^*, \ldots, p_T^*$ and $a_1^*, \ldots, a_T^*$ be the prices and accept decisions from all rounds when the buyer algorithm is $\mathcal{B}^*(\texttt{Phased}, \mathcal{D})$, and let $p_1^i, \ldots, p_T^i$ and $a_1^i, \ldots, a_T^i$ be the price and accept decisions from all rounds when the buyer algorithm is $\mathcal{B}^i$. Recall that the values $v_1, \ldots, v_T$ are drawn independently of seller or buyer behavior. Let t_i^- and t_i^+ be the first and last explore rounds in phase i, respectively. We have

$$\begin{aligned}
&\text{BuyerSurplus}(\texttt{Phased}, \mathcal{B}^*(\texttt{Phased}, \mathcal{D}), \mathcal{D}, T) - \text{BuyerSurplus}(\texttt{Phased}, \mathcal{B}^i, \mathcal{D}, T) \\
&= E\left[\sum_{t=1}^{t_i^- - 1} \gamma^{t-1}(a_t^*(v_t - p_t^*) - a_t^i(v_t - p_t^i))\right] + E\left[\sum_{t=t_i^-}^{t_i^+} \gamma^{t-1}(a_t^*(v_t - p_t^*) - a_t^i(v_t - p_t^i))\right] \\
&\quad + E\left[\sum_{t=t_i^+ + 1}^{T} \gamma^{t-1}(a_t^*(v_t - p_t^*) - a_t^i(v_t - p_t^i))\right] \qquad (5.8) \\
&= E\left[\sum_{t=t_i^-}^{t_i^+} \gamma^{t-1}(a_t^*(v_t - p_t^*) - a_t^i(v_t - p_t^i))\right] + E\left[\sum_{t=t_i^+ + 1}^{T} \gamma^{t-1}(a_t^*(v_t - p_t^*) - a_t^i(v_t - p_t^i))\right] \qquad (5.9) \\
&= E\left[\sum_{t=t_i^-}^{t_i^+} \gamma^{t-1}(a_t^* - a_t^i)(v_t - p_t^i)\right] + E\left[\sum_{t=t_i^+ + 1}^{T} \gamma^{t-1}(a_t^*(v_t - p_t^*) - a_t^i(v_t - p_t^i))\right] \qquad (5.10)
\end{aligned}$$

$$\leq E\left[\sum_{t=t_i^-}^{t_i^+}\gamma^{t-1}(a_t^*-a_t^i)(v_t-p_t^i)\right]+\gamma^{t_i^+}T_\gamma \tag{5.11}$$

$$= \Pr[L_{p,i}\geq C_\delta]E\left[\sum_{t=t_i^-}^{t_i^+}\gamma^{t-1}(a_t^*-a_t^i)(v_t-p_t^i)\;\middle|\; L_{p,i}\geq C_\delta\right]$$
$$+\Pr[L_{p,i}< C_\delta]E\left[\sum_{t=t_i^-}^{t_i^+}\gamma^{t-1}(a_t^*-a_t^i)(v_t-p_t^i)\;\middle|\; L_{p,i}< C_\delta\right]+\gamma^{t_i^+}T_\gamma$$

$$\leq \Pr[L_{p,i}\geq C_\delta]E\left[\sum_{t\in\mathcal{L}_{p,i}}\gamma^{t-1}(a_t^*-a_t^i)(v_t-p_t^i)\;\middle|\; L_{p,i}\geq C_\delta\right]+\gamma^{t_i^+}T_\gamma \tag{5.12}$$

$$\leq \Pr[L_{p,i}\geq C_\delta]E\left[\sum_{t\in\mathcal{L}_{p,i}}\gamma^{t-1}(-\lambda)\;\middle|\; L_{p,i}\geq C_\delta\right]+\gamma^{t_i^+}T_\gamma \tag{5.13}$$

$$\leq \Pr[L_{p,i}\geq C_\delta]\sum_{t=t_i^+-C_\delta+1}^{t_i^+}\gamma^{t-1}(-\lambda)+\gamma^{t_i^+}T_\gamma \tag{5.14}$$

$$< \delta\sum_{t=t_i^+-C_\delta+1}^{t_i^+}\gamma^{t-1}(-\lambda)+\gamma^{t_i^+}T_\gamma \tag{5.15}$$

$$= -\delta\lambda\gamma^{t_i^+-C_\delta}\left(\frac{1-\gamma^{C_\delta}}{1-\gamma}\right)+\gamma^{t_i^+}T_\gamma=\frac{\gamma^{t_i^+}}{1-\gamma}\left(\frac{-\delta\lambda(1-\gamma^{C_\delta})}{\gamma^{C_\delta}}+(1-\gamma)T_\gamma\right)=0 \tag{5.16}$$

Eq. (5.8) follows from the definition of surplus and the linearity of expectation. Eq. (5.9) holds because $\mathcal{B}^*(\texttt{Phased},\mathcal{D})$ and $\mathcal{B}^i$ behave identically before phase i. Eq. (5.10) holds because the prices offered during explore rounds are independent of the buyer's algorithm, and thus $p_t^i = p_t^*$ for $t \in \{t_i^-,\ldots,t_i^+\}$. The fact that $a_t^i = \mathbf{1}\{v_t \geq p_t^i\}$ for $t \geq t_i^-$ implies $a_t^*(v_t-p_t^*)-a_t^i(v_t-p_t^i)\leq 1$ for $t\geq t_i^-$, which yields Eq. (5.11), and also implies $(a_t^*-a_t^i)(v_t-p_t^i)\leq 0$ for $t\geq t_i^-$, which yields Eq. (5.12) (recall that $\mathcal{L}_{p,i}\subseteq\{t_i^-,\ldots,t_i^+\}$). The definition of λ-lies and the fact that $p_t^i = p_t^*$ for $t\in\mathcal{L}_{p,i}$ implies Eq. (5.13). Eq. (5.14) holds because γ^{t-1} is decreasing in t. Eq. (5.15) follows from our assumption that $\Pr[L_{p,i}\geq C_\delta] > \delta$. Eq. (5.16) follows from the definition of C_δ. □

We are ready to prove an upper bound on the regret of the `Phased` algorithm.

Restatement of Theorem 5.2 *Assume* $F(p) \equiv \Pr_{v\sim\mathcal{D}}[v \geq p]$ *is* K*-Lipschitz. Let* $\Delta = \min_{p\in\mathcal{P}\setminus\{p^*\}} p^*F(p^*) - pF(p)$*, where* $p^* = \arg\max_{p\in\mathcal{P}} pF(p)$*. For any parameter* $\alpha \in (0,1)$ *of the* `Phased` *algorithm there exist constants* c_1, c_2, c_3, c_4 *such that*

$$\begin{aligned}\text{Regret}(\texttt{Phased}, \mathcal{D}, T) \leq\ & c_1|\mathcal{P}|T^\alpha + c_2\frac{|\mathcal{P}|^2}{\Delta^{2/\alpha}}(\log T)^{1/\alpha} \\ & + c_3\frac{|\mathcal{P}|^2}{\Delta^{1/\alpha}}T_\gamma^{1/\alpha}(\log T + \log(K/\Delta))^{1/\alpha} + c_4|\mathcal{P}| \\ =\ & \tilde{O}(T^\alpha + T_\gamma^{1/\alpha}).\end{aligned}$$

Proof. Define $n = \lceil \log_2 T \rceil$ and let $\mathcal{T}_i^{\text{explore}}$ and $\mathcal{T}_i^{\text{exploit}}$ be the set of explore and exploit rounds of phase $i \in \{1, \ldots, n\}$. Since the phase n may only be partially completed at the termination of the algorithm we allow $\mathcal{T}_n^{\text{explore}}$ and $\mathcal{T}_n^{\text{exploit}}$ to be partially or completely empty. Note that for the `Phased` algorithm the behavior of a buyer during exploit rounds does not affect the prices offered in future rounds. Since $\tilde{p}_i$ is the price offered in each exploit round of phase i, a surplus-maximizing buyer will choose $a_t = \mathbf{1}\{v_t \geq \tilde{p}_i\}$ in any exploit round t of phase i. So we can upper bound the regret of the `Phased` algorithm in terms of the number of explore rounds and the probability that $\tilde{p}_i \neq p^*$ during exploit rounds. We have

$$\begin{aligned}\text{Regret}(\texttt{Phased}, \mathcal{D}, T) &= E\left[\sum_{t=1}^{T} p^*F(p^*) - a_t p_t\right] \\ &= \sum_{i=1}^{n}\sum_{t\in\mathcal{T}_i^{\text{explore}}} E\left[p^*F(p^*) - a_t p_t\right] + \sum_{i=1}^{n}\sum_{t\in\mathcal{T}_i^{\text{exploit}}} E\left[p^*F(p^*) - a_t p_t\right] \\ &\leq \sum_{i=1}^{n}|\mathcal{P}|S_i + \sum_{i=1}^{n}\sum_{p\in\mathcal{P}\setminus\{p^*\}} \Pr\left[\tilde{p}_i = p\right](T_i - |\mathcal{P}|S_i) \\ &\leq \sum_{i=1}^{n}|\mathcal{P}|S_i + \sum_{p\in\mathcal{P}\setminus\{p^*\}}\sum_{i=1}^{i^*} T_i + \sum_{p\in\mathcal{P}\setminus\{p^*\}}\sum_{i=i^*+1}^{n}\Pr\left[\tilde{p}_i = p\right]T_i \qquad (5.17)\end{aligned}$$

where expectations and probabilities are with respect to value distribution $\mathcal{D}$, seller al-

gorithm `Phased`, and buyer algorithm $\mathcal{B}^*(\texttt{Phased}, \mathcal{D})$. We will now bound each term in Eq. (5.17). Let $\lambda = \frac{\Delta}{8K}$.

Recall that $T_i = 2^i$ and $S_i \leq T_i^\alpha$, which implies $\sum_{i=1}^n S_i \leq \sum_{i=1}^n 2^{\alpha i}$. Since $n \leq \log_2 T + 1$ we have $2^n \leq 2T$. Thus

$$\sum_{i=1}^{n} S_i \leq \sum_{i=1}^{n} 2^{\alpha i} \leq \frac{(2^\alpha)^{n+1} - 1}{2^\alpha - 1} = \frac{(2^{n+1})^\alpha - 1}{2^\alpha - 1} \leq \frac{4^\alpha T^\alpha - 1}{2^\alpha - 1} \leq \frac{4^\alpha}{2^\alpha - 1} T^\alpha. \tag{5.18}$$

where the first inequality follows from the formula for a geometric series (this is just the standard "doubling trick").

By the definition of S_i and i^* we have $T_{i^*-1} < \max(D_T^{1/\alpha}, |\mathcal{P}|D_T) \leq D_T^{1/\alpha} + |\mathcal{P}|D_T$, which implies $T_{i^*+1} \leq 4D_T^{1/\alpha} + 4|\mathcal{P}|D_T$. Also note that $\sum_{j \leq i} T_j \leq T_{i+1}$ for all i, again because $T_i = 2^i$. Thus

$$\sum_{p \in \mathcal{P} \setminus \{p^*\}} \sum_{i \leq i^*} T_i \leq \sum_{p \in \mathcal{P} \setminus \{p^*\}} 4D_T^{1/\alpha} + 4|\mathcal{P}|D_T \leq 4|\mathcal{P}|D_T^{1/\alpha} + 4|\mathcal{P}|^2 D_T \leq 8|\mathcal{P}|^2 D_T^{1/\alpha} \tag{5.19}$$

Finally, for any $p \neq p^*$ and $i > i^*$ if $\tilde{p}_i = p$ then $\tilde{r}_{p^*,i} < \tilde{r}_{p,i}$, which by Lemma 5.7 implies that either the event from Lemma 5.6 does not occur,

$$L_{p,i} \geq \frac{\Delta - 4K\lambda}{4p} S_i, \text{ or} \tag{5.20}$$

$$L_{p^*,i} \geq \frac{\Delta - 4K\lambda}{4p^*} S_i. \tag{5.21}$$

Since $\lambda = \frac{\Delta}{8K}$ and $p, p^* \leq 1$, Eq. (5.20) and Eq. (5.21) respectively imply

$$L_{p,i} \geq \frac{\Delta}{8} S_i, \text{ or} \tag{5.22}$$

$$L_{p^*,i} \geq \frac{\Delta}{8} S_i. \tag{5.23}$$

The event from Lemma 5.6 (call it event $\neg A$) occurs with probability at least $1 - 2T^{-1}$. And since $S_i \geq D_T \geq (8/\Delta) C_{\frac{1}{T}}$ for all $i \geq i^*$, we have that Eq. (5.22) and Eq. (5.23) imply

either $L_{p,i} \geq C_{\frac{1}{T}}$ (call it event B_1) or $L_{p^*,i} \geq C_{\frac{1}{T}}$ (call it event B_2), which by Lemma 5.8 each occur with probability at most T^{-1} assuming the event $\neg A$ has occurred. Combining these results, we have

$$\begin{aligned}\Pr[\tilde{p}_i = p] &\leq \Pr[A \vee B_1 \vee B_2] \\ &\leq \Pr[A] + \sum_{i=1,2} \Pr[B_i|A] \Pr[A] + \Pr[B_i|\neg A] \Pr[\neg A] \\ &\leq 2T^{-1} + 2(2T^{-1} + T^{-1}) = 8T^{-1}\,,\end{aligned}$$

and therefore

$$\sum_{p \in \mathcal{P} \setminus \{p^*\}} \sum_{i > i^*} \Pr\left[\tilde{p}_i = p\right] T_i \leq 8|\mathcal{P}| \tag{5.24}$$

Combining Eqs. (5.18), (5.19) and (5.24) with Eq. (5.17) yields

$$\text{Regret}(\texttt{Phased}, \mathcal{D}, T) \leq \frac{4^\alpha}{2^\alpha - 1}|\mathcal{P}|T^\alpha + 8|\mathcal{P}|^2 D_T^{1/\alpha} + 8|\mathcal{P}|$$

Plugging in the definitions $D_T = \max\left(\frac{16}{\Delta^2}\log T, \frac{8}{\Delta}C_{\frac{1}{T}}\right)$ and $\lambda = \frac{\Delta}{8K}$, we have

$$\begin{aligned}\text{Regret}(\texttt{Phased}, \mathcal{D}, T) \;\leq\; & \frac{4^\alpha}{2^\alpha - 1}|\mathcal{P}|T^\alpha + 8|\mathcal{P}|^2 \left(\frac{16}{\Delta^2}\log T\right)^{1/\alpha} \\ & + 8|\mathcal{P}|^2 \left(\frac{8}{\Delta}\log\left(1 + \frac{8K(1-\gamma)T_\gamma T}{\Delta}\right)\log\left(\frac{1}{\gamma}\right)^{-1}\right)^{1/\alpha} \\ & + 8|\mathcal{P}|. \end{aligned} \tag{5.25}$$

Suppose γ and T satisfy $\gamma^T \geq 1/2$. Then $\gamma^t \geq 1/2$ for all $t \leq T$, and furthermore $T_\gamma = \sum_{t=1}^T \gamma^{t-1} \geq T/2$. Since $\text{Regret}(\texttt{Phased}, \mathcal{D}, T) \leq T$ holds trivially, we have

$$\text{Regret}(\texttt{Phased}, \mathcal{D}, T) \leq T \leq 2T_\gamma \leq T^\alpha + 2T_\gamma^{1/\alpha},$$

satisfying the theorem. Therefore, we assume that $\gamma^T \leq 1/2$. Since $T_\gamma = \sum_{t=1}^T \gamma^{t-1} = \frac{1-\gamma^T}{1-\gamma}$

we have

$$2T_\gamma = 2\left(\frac{1-\gamma^T}{1-\gamma}\right) \geq \frac{1}{1-\gamma} \geq \frac{1}{\log(1/\gamma)}$$

where the first inequality follows from $2(1-\gamma^T) \geq 1$ and the second inequality follows from $x \geq \log(1+x)$ for all x (just substitute $x = \gamma - 1$ and rearrange). Thus we can upper bound $\log\left(\frac{1}{\gamma}\right)^{-1}$ in Eq. (5.25) by $2T_\gamma$, and simplifying yields the statement of the theorem. □

5.6.3. Lower Bound Proofs

Restatement of Lemma 5.3 *For any incentive compatible and rational strategy $\mathcal{S}$ there exists $v \in [0,1]$ such that* $\text{SSRegret}(\mathcal{S}, v) \geq \frac{1}{12}$.

Proof. Fix a incentive compatible and rational strategy $\mathcal{S}$. Let $\text{SellerRevenue}(b) = E_{(p,a)\sim\mathcal{S}_b}[ap]$ be the seller's expected revenue if the buyer bids b, and let $\text{BuyerSurplus}(b, v) = E_{(p,a)\sim\mathcal{S}_b}[a(v-p)]$ be the buyer's expected surplus if she bids b and her value is v. It suffices to show that there exists $v \in [0,1]$ such that $v - \text{SellerRevenue}(v) \geq \frac{1}{12}$.

Before proceeding, we establish some properties of $\mathcal{S}$. Incentive compatibility of $\mathcal{S}$ ensures that

$$\text{BuyerSurplus}(v, v) \geq \text{BuyerSurplus}(b, v) \tag{5.26}$$

for all $b, v \in [0,1]$, and rationality of $\mathcal{S}$ ensures that

$$\text{BuyerSurplus}(v, v) \geq 0 \tag{5.27}$$

for all $v \in [0,1]$. Also

$$\text{SellerRevenue}(b) + \text{BuyerSurplus}(b, v) = E_{(p,a)\sim\mathcal{S}_b}[a]v \tag{5.28}$$

for all $b, v \in [0,1]$, which follows directly from definitions, and

$$\text{SellerRevenue}(v) \leq E_{(p,a)\sim\mathcal{S}_v}[a]v \tag{5.29}$$

for all $v \in [0,1]$, which follows from rationality: By (5.28) we have $\text{BuyerSurplus}(v,v) = E_{(p,a)\sim\mathcal{S}_v}[a]v - \text{SellerRevenue}(v)$, and thus if (5.29) were false we would have $\text{BuyerSurplus}(v,v) < 0$, which contradicts (5.27).

Now observe that for any $b, v \in [0,1]$

$$\begin{aligned}
v - \text{SellerRevenue}(v) &\geq E_{(p,a)\sim\mathcal{S}_v}[a]v - \text{SellerRevenue}(v) \\
&= \text{BuyerSurplus}(v,v) && (5.30) \\
&\geq \text{BuyerSurplus}(b,v) && (5.31) \\
&= E_{(p,a)\sim\mathcal{S}_b}[a(v-p)] \\
&= E_{(p,a)\sim\mathcal{S}_b}[a]v - E_{(p,a)\sim\mathcal{S}_b}[ap] \\
&= E_{(p,a)\sim\mathcal{S}_b}[a]v - \text{SellerRevenue}(b) \\
&\geq \left(\frac{\text{SellerRevenue}(b)}{b}\right) v - \text{SellerRevenue}(b) && (5.32) \\
&= (v-b)\left(\frac{\text{SellerRevenue}(b)}{b}\right)
\end{aligned}$$

where (5.30) follows from (5.28), (5.31) follows from (5.26), and (5.32) follows from (5.29). Now let $b = \frac{1}{4}$ and $v = \frac{1}{2}$. If $v - \text{SellerRevenue}(v) \geq \frac{1}{6}$ we are done. Otherwise the first and last lines from the above chain of inequalities and $v - \text{SellerRevenue}(v) < \frac{1}{6}$ imply

$$\frac{\text{SellerRevenue}(b)}{b} \leq \frac{v - \text{SellerRevenue}(v)}{v-b} < \frac{1}{6}\frac{1}{v-b} = \frac{2}{3}$$

which can be rearranged into $b - \text{SellerRevenue}(b) \geq \frac{1}{3}b \geq \frac{1}{12}$. □

Restatement of Lemma 5.4 *Let the buyer's discount sequence $\{\gamma_t\}$ be positive and non-increasing. For any seller algorithm $\mathcal{A}$, value distribution $\mathcal{D}$, and surplus-maximizing buyer algorithm $\mathcal{B}^*(\mathcal{A},\mathcal{D})$, $E\left[\sum_{t=1}^{T}\gamma_t a_t(v_t - p_t)\right] \leq E\left[\sum_{t=1}^{T} a_t(v_t - p_t)\right]$*

Proof. It will be convenient to define the following (all expectations in these definitions are

with respect to $\mathcal{A}, \mathcal{D}$ and $\mathcal{B}^*(\mathcal{A}, \mathcal{D})$):

$$\begin{aligned}
\text{rev}(t_1, t_2) &= E\left[\sum_{t=t_1}^{t_2} a_t p_t\right] \\
\text{sur}(t_1, t_2) &= E\left[\sum_{t=t_1}^{t_2} \gamma_t a_t (v_t - p_t)\right] \\
\text{udsur}(t_1, t_2) &= E\left[\sum_{t=t_1}^{t_2} a_t (v_t - p_t)\right] \\
\text{totval}(t_1, t_2) &= E\left[\sum_{t=t_1}^{t_2} a_t v_t\right]
\end{aligned}$$

where "udsur" stands for "undiscounted surplus" and "totval" stands for "total value". Note that by definition

$$\text{rev}(t_1, t_2) + \text{udsur}(t_1, t_2) = \text{totval}(t_1, t_2). \tag{5.33}$$

Also, since $\mathcal{B}^*(\mathcal{A}, \mathcal{D})$ is a surplus-maximizing buyer strategy, $\text{sur}(t, T) \geq 0$ for all rounds t, because otherwise the buyer could increase her surplus by following $\mathcal{B}^*(\mathcal{A}, \mathcal{D})$ until round $t-1$ and then selecting $a_{t'} = 0$ for all rounds $t' \geq t$.

We will first prove that $\text{sur}(t, T) \leq \gamma_t \text{udsur}(t, T)$ for all rounds t. The proof will proceed by induction. For the base case, we have $\text{sur}(T, T) = \gamma_T \text{udsur}(T, T)$ by definition. Now assume for the inductive hypothesis that $\text{sur}(t+1, T) \leq \gamma_{t+1} \text{udsur}(t+1, T)$. Since $\text{sur}(t+1, T) \geq 0$ and $\gamma_{t+1} > 0$, by the inductive hypothesis we have $\text{udsur}(t+1, T) \geq 0$. Therefore

$$\begin{aligned}
\text{sur}(t, T) &= \text{sur}(t, t) + \text{sur}(t+1, T) \\
&= \gamma_t \text{udsur}(t, t) + \text{sur}(t+1, T) \\
&\leq \gamma_t \text{udsur}(t, t) + \gamma_{t+1} \text{udsur}(t+1, T) && (5.34) \\
&\leq \gamma_t \text{udsur}(t, t) + \gamma_t \text{udsur}(t+1, T) && (5.35) \\
&= \gamma_t \text{udsur}(t, T)
\end{aligned}$$

where Eq. (5.34) follows from the inductive hypothesis and Eq. (5.35) follows because $\text{udsur}(t+1,T) \geq 0$ and $\gamma_t \geq \gamma_{t+1}$. Thus $\text{sur}(t,T) \leq \gamma_t \text{udsur}(t,T)$.

Since $\text{sur}(1,T) \leq \gamma_1 \text{udsur}(1,T)$ and $\gamma_1 \leq 1$, by Eq. (5.33) we have $\text{rev}(1,T) + \text{sur}(1,T) \leq \text{totval}(1,T)$, which proves the lemma. □

CHAPTER 6 : Advertiser Budget Optimization in Keyword Auctions

We conclude with an experimental study of algorithms for optimized budget expenditure in sponsored search. Given an advertiser's budget, the goal of such algorithms is to maximize the number of clicks obtained during each budgeting period. We consider a single-slot model in which an algorithm's competing bid — representing the rest of the "market" for clicks — is drawn from a fixed and unknown probability distribution. While in reality, advertisers (or their proxies) may often bid strategically and not stochastically, we view this assumption as analogous to classical models in finance, where despite strategic behavior of traders at the individual level, models of macroscopic price evolution that are stochastic (such as Brownian motion models) have been quite effective in developing both models and algorithms. Our empirical results will demonstrate that algorithms designed for these stochastic assumptions also perform quite well on the non-stochastic sequence of bids actually occurring in real search auctions.

The assumption of stochastic bids by the competing market leads to a Markov Decision Process (MDP) formulation of the optimal policy, where the states of the MDP specify the remaining time in the period and the remaining budget. However, the second-price nature of sponsored search introduces the challenge of *censored* observations: only if we win the click do we observe the actual competing price; otherwise, we only know our bid was too low.

Our main contributions are the introduction of efficient algorithms that combine the MDP formulation with the classical Kaplan-Meier Kaplan and Meier (1958) or product-limit estimator for censored observations, and a large-scale empirical demonstration that these algorithms are extremely effective in practice — even when our underlying distributional assumptions are badly violated. Our source of data is auction-level observations on hundreds of high-volume key-phrases from Microsoft adCenter. We show that our algorithms rapidly learn to compete with the strongest possible benchmark — the performance of an offline-

optimal algorithm that knows the future competing bids, and always selects the cheapest clicks in each period.

6.1. Preliminaries

The optimization problem we consider occurs over a series of *periods*. At the beginning of each period, the budget optimization algorithm is allocated a fresh budget B. This assumption is meant to reflect the manner in which advertisers actually specify their budgets in real sponsored search markets. Broadly speaking, the algorithm's goal is to maximize the number of clicks purchased, in each period, using the budget B.

Each period consists of a number of *auctions*, or an opportunity to earn a click. We consider the *single-slot* setting, wherein the search engine displays a single advertisement for each auction [1]. The algorithm places a *bid* in each auction. Before the bid is placed, a *price* is fixed by the auction mechanism. If the algorithm's bid exceeds this price, the algorithm wins an *impression*. For simplicity, we begin by assuming that winning an impression automatically guarantees that the algorithm also wins a click; we will describe later how to relax this assumption.

Once the algorithm wins a click, it is charged the price from its budget. Otherwise, the algorithm maintains its budget, and the next auction occurs. In actual sponsored search markets, the price is determined by the bids of arbitrary agents competing for the click in a modified second-price auction (see e.g. Varian (2007)).

The major assumption of this work is to instead model the prices as i.i.d. draws from an unknown distribution; we will refer to this price as the *market price*, since it represents the aggregate behavior of our algorithm's competitors. As in other financial applications, it is often analytically intractable to model the individual agents in a market strategically, so we instead consider the market stochastically as a whole. We will demonstrate that our algorithm outperforms other methods in practice on actual auction data from Microsoft

[1] We suspect our methods can be adapted to the multi-slot case, but leave it to future work.

adCenter, even when the i.i.d. assumption is badly violated.

We will consider some fixed, unknown, distribution $\mathcal{P}$ supported on $\mathbb{Z}^+$ with mass function $p(\cdot)$. On each auction, the market price is an independent random variable distributed as $\mathcal{P}$. We think of the budget B and market prices as being expressed in terms of the smallest unit of currency that can be bid by the algorithm. Modeling the problem in this manner motivates a natural algorithm.

It is important to note that the market prices are not observed directly by the algorithm. Rather, the algorithm is only privy to the consequences of its bid (whether a click or impression is received), and changes to its budget.

Succinctly, we consider the following protocol:

for period $u = 1, 2, ...$ **do**
  $B_{u,T} = B$
  for auctions remaining $t = T, T-1, ..., 1$ **do**
    Algorithm bids $b_{u,t} \leq B_{u,t}$.
    Nature draws price $x_{u,t} \sim \mathcal{P}$.
    if $b_{u,t} \geq x_{u,t}$ **then**
      $c_{u,t} \leftarrow 1$
      $B_{u,t-1} \leftarrow B_{u,t} - x_{u,t}$
    else
      $c_{u,t} \leftarrow 0$
      $B_{u,t-1} \leftarrow B_{u,t}$
    end if
    Algorithm observes $c_{u,t}$, $B_{u,t-1}$
  end for
end for

When an algorithm places a large enough a bid, winning the click, it also observes the true

market price, since its budget it reduced by that amount. However, should the algorithm fail to win the click, it only knows that the market bid was higher than the bid placed. Thus, the algorithm receives what is known in the statistical literature as partially *right-censored observations* of the market prices $\{x_{u,t}\}$.

Informally, we always assume that an algorithm has available to it any information it would have in a real sponsored search auction (although not always at the same granularity), and no more. So it may be informed of whether it received a click or impression on an auction-by-auction basis, but not information regarding prices if it did not win the click.

Finally, we assume there is only a single keyword which the advertiser is bidding on. Our methods generalize to the setting where there are multiple keywords with multiple click-through rates and valuations for a click. However, for simplicity, we do not consider these extensions in this work.

6.1.1. Notation

Given a distribution $\mathcal{P}$, supported on $\mathbb{Z}^+$, with mass function p, we let the tail function $T_p(b) = \sum_{b'=b+1}^{\infty} p(b')$ denote the mass to the right of b.

We use $[N]$ to mean the set $\{1, ..., N\}$, and $[N]_0 = [N] \cup \{0\}$.

6.2. Related Work

There is some prior work directly concerning the problem of optimizing an advertiser's budget Kitts and Leblanc (2004); Zhou et al. (2008), as well as work concerned with characterizing the dynamics or equilibria of a market in which advertisers play from a family of optimizing strategies Borgs et al. (2007); Cary et al. (2007). All these works attempt to model the strategic behavior of agents participating in a sponsored search auction. We will depart from this, modeling the auction market stochastically, as is more common in finance.

The sponsored search budget optimization problem has also been formulated as an instance

of online knapsack Zhou et al. (2008). For the online knapsack problem, it is known that no online algorithm can converge to the optimum in the worst case Marchetti-Spaccamela and Vercellis (1995). The stochastic knapsack problem has also been studied, and there is an algorithm with near-optimal average-case performance Lueker (1995). One of our proposed algorithms is a censored learning version of such an algorithm. Our main proposed algorithm is closely modeled on an algorithm from a financial optimization problem Ganchev et al. (2010), which similarly integrates a censored estimation step with greedy optimization.

Finally, we apply classical techniques from reinforcement learning in a finite state MDP, including Q-learning (c.f. Watkins and Dayan (1992)), as well as classical techniques from the study of censored observations Kaplan and Meier (1958); Turnbull (1974).

6.3. MDP Formulation

An algorithm for the optimization problem introduced in the previous section can be described as an agent in a Markov Decision Process (MDP). An MDP $\mathcal{M}$ can be written as $\mathcal{M} = (\mathcal{S}, \{A_s\}_{s\in\mathcal{S}}, \mu, r)$ where $\mathcal{S}$ is a set of states, and A_s are the set of actions available to the agent in each state s, and $\mathcal{A} = \cup_{s\in\mathcal{S}} A_s$. For $a \in A_s$, $\mu(a, s, s')$ is the probability of transitioning from state s to state s' when taking action a in state s. $r(a, s, s')$ is the expected reward received after taking action a in state s and transitioning to state s'. The goal of the agent is to maximize the expected reward received while transitioning through the MDP.

In our case, the state space is given by $\mathcal{S} = [B]_0 \times [T]_0$. With t auctions remaining in period u, the algorithm is in state $(B_{u,t}, t) \in \mathcal{S}$. Furthermore, the actions available to any algorithm in such a state are the set of bids that are at most $B_{u,t}$. So for any $(b, t) \in \mathcal{S}$ we let $\mathcal{A}_{(b,t)} = [b]_0$.

When $t \geq 1$, two types of transitions are possible. The agent can transition from (b, t) to $(b, t-1)$ or from (b, t) to $(b', t-1)$ where $0 \leq b' < b$. In the former case, the agent must place a bid lower than the market price. Therefore, we have that $\mu(a, (b, t), (b, t-1)) = T_p(a)$.

Furthermore, the agent does not win a click in this case, and $r(a,(b,t),(b,t-1))=0$. In the latter case, let $\delta = b-b'$. The agent must bid at least δ, and the market price must be exactly δ on auction t of the period in question. Therefore, we have that $\mu(a,(b,t),(b',t-1)) = p(\delta)$ and $r(a,(b,t),(b',t-1))=1$ so long as $a \geq \delta$.

When $t=0$, for any action, the agent simply transitions to (B,T) with probability 1, with no reward. The agent's budget is refreshed, and the next period begins.

All other choices for $(a,s,s') \in \mathcal{A}_s \times \mathcal{S} \times \mathcal{S}$ represent invalid moves, and hence $\mu(a,s,s') = r(a,s,s') = 0$.

Finally, conditioned on an agent's choice of action a and current state s, its next state s is independent of all previous actions and states, since the market prices $\{x_{u,t}\}$ are independent. The Markov property is satisfied, and we indeed have an MDP.

We call this the *Sponsored Search MDP* (SS-MDP). If π is a fixed mapping from $\mathcal{S}$ to $\mathcal{A}$ satisfying $\pi(s) \in \mathcal{A}_s$, we say that π is a *policy* for the MDP.

Note that an agent in the SS-MDP, started in an arbitrary state (b,t), arrives at the state (B,T) after exactly $t+1$ actions. Therefore, we can define the random variable $C_\pi(b,t)$ to be the total reward (i.e. number of clicks) attained by policy π before returning to (B,T). We say that π is *an optimal policy for the SS-MDP* if an agent started at (B,T), playing $\mu(s)$ in each state s encountered, maximizes the expected number of clicks rewarded before returning to (B,T). In other words:

Definition 6.1. *A policy π^* is an optimal policy for the SS-MDP if $\pi^* \in \arg\max_\pi E[C_\pi(B,T)]$.*

The SS-MDP is determined by the choice of budget B, time T and distribution p. We will want to make the optimal policy's dependence on p explicit, and consequently we will write it a π^*_p.

6.4. The Value Function

MDPs lend themselves to dynamic programming. Indeed, we can characterize exactly the optimal policy for the SS-MDP when the probability mass function p is known.

For a distribution p, let $V_p(b,t)$, the *value function for* p, denote the expected number of clicks received by an optimal policy started at state (b,t). That is, if π_p^* is an optimal policy, then $V_p(b,t) = E[C_{\pi^*}(b,t)]$.

First note that when $T = 0$, $V_p(B,T) \equiv 0$. Consider the policy $\pi^*_{a,b,t}$ that takes action a in state (b,t), and plays optimally thereafter. Let $V_p(a,b,t) = E[C_{\pi^*_{a,b,t}}(b,t)]$ be the clicks received by such a policy from state (b,t). Observe that $V_p(a,b,t)$ can be written in terms of $V_p(\cdot, t-1)$:

$$V_p(a,b,t) = \sum_{\delta=1}^{a} p(\delta)[1 + V_p(b-\delta, t-1)] + T_p(a)V_p(b,t-1).$$

In other words, if the market price is $\delta \leq a$, which occurs with probability $p(\delta)$, the agent will win a click at the price of δ and transition to state $(b-\delta, t-1)$. At this point it behaves optimally, earning $V_p(b-\delta, t-1)$ clicks in expectation. If the market price is greater than a, which occurs with probability $T_p(a)$, the agent will retain its budget, transitioning to the state $(b, t-1)$, earning $V_p(b,t-1)$ clicks in expectation. Furthermore, we know that:

$$V_p(b,t) = \arg\max_{a \leq b} V_p(a,b,t).$$

Therefore, if p and $V_p(\cdot, T-1)$ are known then we can compute $V_p(a,b,t)$ in $O(B)$ operations, and so compute $V_p(b,t)$ in $O(B)$ operations. Recalling that $V_p(b,0) \equiv 0$, we can compute $V_p(b,t)$ for all $(b,t) \in [B]_0 \times [T]_0$ in $O(B^2T)$ operations.

6.5. Censored Data

In the previous sections, we described how to compute the optimal policy π_p^* when p is known. A natural algorithm for budget optimization is therefore to maintain an estimate $\hat{p}$

of p, and bid greedily according to π^*_p. Before describing such an algorithm, we will discuss the problem of estimating p.

As introduced in Section 6.1, the observations received by a budget-optimization algorithm are partially right-censored data. We begin with a general discussion of censoring.

Suppose that $\mathcal{P}$ is a distribution with mass function p and $(z_1, ..., z_n)$ are i.i.d., $\mathcal{P}$-distributed random variables. Fix n integers $k_1, ..., k_n$, and define $o_i = \min(z_i, k_i)$. We say that the sample $\{o_i\}$ is partially right-censored data.

If $o_i < k_i$, we say that o_i is a direct observation. In other words, $o_i = z_i$ and we have observed the true value of z_i. Otherwise, $o_i = k_i$ and we say that o_i is a censored observation. We know only that $z_i \geq k_i$.

Given such partially right-censored data, the Product-Limit estimator Kaplan and Meier (1958) is the non-parametric maximum-likelihood estimator for p.

Definition 6.2. *Let $\mathcal{P}$ be a discrete distribution with mass function p, and let $\{z_i\}$ be i.i.d. $\mathcal{P}$-distributed random variables. Given integers $K = (k_1, ..., k_n)$ and observations $O = (o_1, ..., o_n)$ where $o_i = \min(z_i, k_i)$, let $PL(K, O)$ be the Product-Limit estimator for p.*

Specifically, given integers K, and a set of observations O generated by a distribution $\mathcal{P}$, let $D(s) = |\{o_i \in O \mid s = o_i < k_i\}|$ be the number of direct observations of value s, and $N(s) = |\{o_i \in O \mid s \leq o_i, s < k_i\}|$. Now let $S(t) = \prod_{s=1}^{t-1} 1 - \frac{D(s)}{N(s)}$. The CDF of $PL(K, O)$ is given by $1 - S(t)$.

In our setting, we are receiving censored observation of the random variables $\{x_{u,t}\}$ where the censoring set K is given by $\{b_{u,t} + 1\}$, and $o_{u,t} = \min\{b_{u,t} + 1, x_{u,t}\}$. When a click is received (i.e. $x_{u,t} < b_{u,t} + 1$), we observe $x_{u,t}$ directly since $x_{u,t} = B_{u,t} - B_{u,t-1}$, the amount which the algorithm is charged for winning the click. Otherwise, the algorithm is charged nothing, and we only know that $x_{u,t} \geq b_{u,t} + 1$.

Finally, we will eventually consider the setting in which winning an impression does not

necessarily guarantee winning a click. In such a setting, we will have both left-censored and right-censored observations of $x_{u,t}$, what is known as *doubly-censored data*.

If the algorithm bids $b_{u,t}$ and does not win the impression, we know that $x_{u,t} \geq b_{u,t} + 1$. Similarly, if the algorithm wins both the impression and the click, it gets to observe $x_{u,t}$ directly. However, should the algorithm win the impression but not the click, it is only informed that it placed a large-enough bid (that it won the impression), or $x_{u,t} < b_{u,t} + 1$, without observing $x_{u,t}$ directly. We give a more detailed discussion of this setting in Section 6.8. For doubly-censored data, there is algorithm giving the non-parametric MLE Turnbull (1974).

6.6. Greedy Product-Limit Algorithm

The algorithm we propose maintains an estimate $\hat{p}$ of p. With budget b remaining, and t auctions remaining, the algorithm will greedily use its current estimate of p, and bid $\pi_{\hat{p}}(b, t)$. The pseudo-code for *Greedy Product-Limit* contains a detailed description.

Algorithm 6 Greedy Product-Limit

Input: Budget B

Initialize distribution $\hat{p}$ uniform on $[B]$
Initialize $K = []$; Initialize $O = []$
for period $u = 1, 2, ...$ **do**
    Set $B_{u,T} := B$
    for auctions remaining t=T,T-1,...,1 **do**
        Bid $\pi^*_{\hat{p}}(B_{u,t}, t)$
        Set $k_{u,t} \leftarrow \pi^*_{\hat{p}}(B_{u,t}, t) + 1$
        $K \leftarrow [K, k_{u,t}]$
        if Click won at price $x_{u,t}$ **then**
            $O \leftarrow [O, x_{u,t}]$
        else
            $O \leftarrow [O, k_{u,t}]$
        end if
        Update $\hat{p}$ to $PL(K, O)$
    end for
end for

6.7. Competing Algorithms

In this section we will describe a few alternative strategies against which we compare *Greedy Product-Limit*. The first relies on an observation that, given an *arbitrary* sequence of market prices, there is a simple bidding strategy that has a constant competitive ratio to the offline optimal.

6.7.1. Offline Optimality

So far we have focused our attention on the notion of optimality introduced in Section 6.3. Namely, an algorithm is optimal if it achieves $V_p(B,T)$ clicks, in expectation, in every period. However, given an arbitrary (non-stochastic) vector of T market prices $\mathbf{x} = (x_1, ..., x_T)$, we can define $C^*(\mathbf{x}, B)$ to be the maximum number of clicks that could be attained by any sequence of bids, knowing $\mathbf{x}$ *a priori*. In other words, if $\mathbf{b} \in \{0,1\}^T$, and $\|\mathbf{b}\|_0 = |\{b_i \mid b_i = 1\}|$, then we define

Definition 6.3.

$$C^*(\mathbf{x}, B) \triangleq \max_{\mathbf{b}\in\{0,1\}^T} \|\mathbf{b}\|_0 \textit{ subject to } \mathbf{x}\cdot\mathbf{b} \leq B.$$

We call a sequence of bids for $\mathbf{x}$ that attains $C(\mathbf{x}, B)$ clicks an *optimal offline policy*. Notice that one attains the optimal offline policy by greedily selecting to win the clicks with the cheapest prices, until the budget B is saturated.

6.7.2. Fixed Price

Competing against the notion of optimality introduced in Section 6.7.1 may seem onerous in the online setting. Indeed, competing against an arbitrary sequence of prices is a special case of the online knapsack problem, which is known to be hard. However, we will now show that for any sequence of prices $\mathbf{x}$, there always exists a simplistic bidding policy which would have attained a constant factor of the bids of the optimal offline policy.

Let $Fixed(b)$ be the policy that bids b on every auction that it has budget to do so, and

define $C(\mathbf{x}, b, B)$ to be the number of clicks attained by $Fixed(b)$ against $\mathbf{x}$ with budget B.

Theorem 6.1. *For any sequence of prices* $\mathbf{x}$*, and budget* B*, there exists a bid* b *such that* $C(\mathbf{x}, b, B) \geq \frac{1}{2}C^*(\mathbf{x}, B)$.

Proof. Let b^* be the value of the price for the most expensive click that the optimal offline policy selects to win. Suppose that the optimal offline policy wins $M + N$ clicks, where M clicks were won with a price of exactly b^* and the remaining N clicks were won with a price of $b^* - 1$ or less.

If $N \geq \frac{1}{2}C^*(\mathbf{x}, B)$, then $Fixed(b^* - 1)$ would win all N clicks, giving the desired result.

Otherwise, we know that $M \geq \frac{1}{2}C^*(\mathbf{x}, B)$. Consider the policy $Fixed(b^*)$. In the worst case, the policy will win only clicks with price b^* before saturating its budget. However, we know that $Mb^* \leq B$, and so $C(\mathbf{x}, b^*, B) \geq M \geq \frac{1}{2}C^*(\mathbf{x}, B)$, as desired.

□

6.7.3. Fixed-Price Search

This motivates a simple algorithm which attempts to find the best fixed-price, *Fixed-Price Search.*

Algorithm 7 Fixed-Price Search

```
1: Select b_1 arbitrarily.
2: for period u = 1, 2, ... do
3:     C_u := 0
4:     for auctions remaining t = T, T − 1, ..., 1 do
5:         Bid b_u
6:         if Click won then
7:             C_u ← C_u + 1
8:         end if
9:     end for
10:    b_{u+1} ← UpdateBid({b_{u'}, C_{u'}}_{u'=1}^{u})
11: end for
```

The algorithm plays a fixed-price strategy each period. At the end of the period it uses

a subroutine *UpdateBid* to select a new fixed-price according to how many clicks it has received. There are many reasonable ways to specify the *UpdateBid* subroutine, including using additive or multiplicative updates (e.g. treating each price as an expert and running a bandit algorithm such as Exp3 Auer et al. (2003)). But general, the performance of any such approach cannot overcome the fixed-price "gap" of $E_{\mathbf{x}}[C^*(\mathbf{x}, B) - \max_b C(\mathbf{x}, b, B)]$.

6.7.4. Q-learning

Given the MDP formulation of the problem in Section 6.3, we may hope to solve the problem using techniques from reinforcement learning. Q-learning with exploration is one of the simplest algorithms for reinforcement learning, giving good results in a number of applications.

In Q-learning, the agent begins with an estimate $Q(a, b, t)$ of the function $V_p(a, b, t)$, called the Q-value, for each state (b, t) and action $a \in [b]$. In a state (b, t), the agent greedily performs the best action a^* for that state using the current Q-values receiving some reward $\hat{r}$ and arriving at a new state $(b', t-1)$. The Q-values for state $(b', t-1)$ and the observed reward $\hat{r}$ are then used to update $Q(a^*, b, t)$. This is often combined with forced exploration. Notice that Q-learning will necessarily ignore the special assumptions placed on the underlying MDP. In particular, from our discussion in Section 6.3, we have that $\pi(a, (b_1, t_1), (b'_1, t_1 - 1)) = \pi(a, (b_2, t_2), (b'_2, t_2 - 1))$ when $b_1 - b'_1 = b_2 - b'_2$.

6.7.5. Knapsack Approaches

As referenced in Section 6.2, the problem of budget optimization in sponsored search is very related to the online knapsack problem. In the online knapsack problem, an optimizer is presented with a sequence of items with values and weights. At each time step, the optimizer makes an irrevocable decision to take the item (subtracting its weight from the optimizer's budget, and gaining its value). In the worst case, Marchetti-Spaccamela et al. demonstrate that a constant-factor competitive ratio with the offline is not possible Marchetti-Spaccamela and Vercellis (1995). Nevertheless, there are many results from the

online knapsack literature that are applicable to our setting.

While in the worst case the online knapsack problem is hard, Lueker gives an average-case analysis for an algorithm for the *Stochastic Knapsack Problem*, which is related to our setting Lueker (1995). In the Stochastic Knapsack Problem, items $\{(r_i, x_i)\}$ are i.i.d. draws from some fixed, *known*, distribution. r_i is the profit or reward earned by taking the item, and x_i is the price of the item. In our setting, all clicks are considered indistinguishable for a fixed keyword, and so $r_i = 1$.

Note that the protocol differs from ours in a few ways. Firstly, there is no learning. The underlying distribution is assumed to be known. Secondly, there is no censoring of data, or any notion of an auction. The optimizer is presented with each item up-front, at which point it must make a decision before moving on to the next. Thirdly, in the language of our setting, there is only a single period.

Under these assumptions, the algorithm of Lueker gives a simple algorithm which differs from the true optimum by an average of $\Theta(T)$, where T is the length of the period, assuming that the budget available scales with T Lueker (1995).

Nevertheless, the same ideas behind Greedy Product-Limit give us a natural adaptation of this algorithm to our setting. Suppose that the prices are presented up-front and that $\mathcal{P}$ is known. With budget B remaining and time T remaining in a period, the algorithm computes:

$$v(B/T) \triangleq \max_v \{v \mid \sum_{a=1}^{v} a \cdot p(a) \leq B/T\}$$

and takes the click iff its market price x satisfies $x \leq v(B/T)$. Thus, when the prices are not presented up-front, it is equivalent to simply bidding $v(B/T)$.

Note that bidding $v(B/T)$ is natural; it is the bid that, in expectation, costs B/T, or

smooths the remaining budget over the time remaining. We can now combine this with an estimation step, using the product-limit estimator, as we did for *Greedy Product-Limit*, simply replacing line 7 with the assignment $k_{u,t} := v(B/T)$. We refer to this strategy as *LuekerLearn*.

Notice, however, that once $\hat{p}$ has converged to the true p, we should not expect this algorithm to outperform *Greedy Product-Limit*, which would bid optimally. For a fixed choice of distribution p, budget B, and number of auctions T, let $L_p(B,T)$ be the expected number of clicks earned by running the algorithm of Lueker, knowing p. $L_p(B,T) \leq V_p(B,T)$, by definition of $V_p(B,T)$. We will comment (as established by Lueker) that the gap $V_p(B,T) - L_p(B,T)$ is exacerbated by distributions with large variance relative to B and T, an issue we will return to in Section 6.8; see also Figure 4.

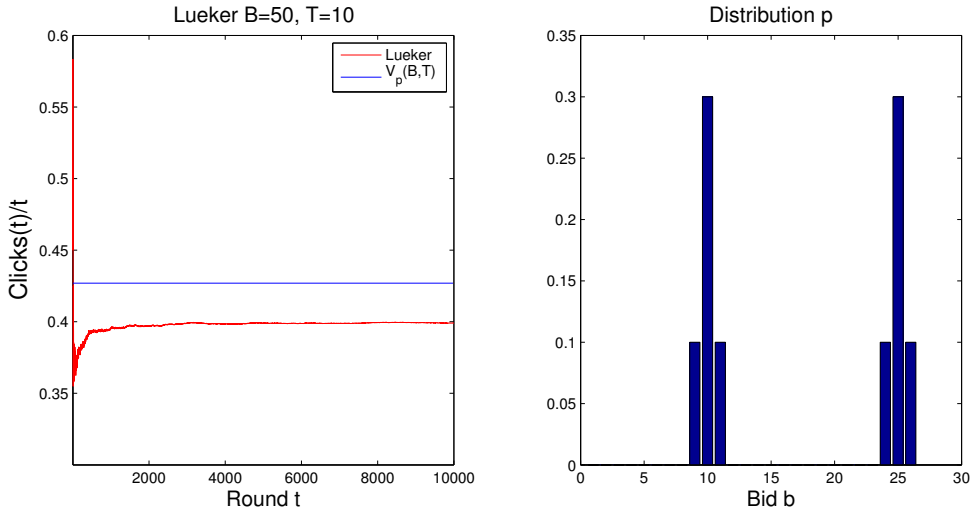

Figure 4: A distribution for which $L_p(B,T)$ is bounded away from $V_p(B,T)$. The red curve plots the performance (averaged over auctions) of the algorithm of Lueker's algorithm when the distribution p is known. The distribution is displayed on the right.

6.7.6. Budget Smoothing

There is also literature in which other budget-smoothing approaches are considered. Zhou et al. consider the budget optimization problem in sponsored search as a online knapsack

problem directly Zhou et al. (2008). In our setting, their algorithm guarantees a $\ln(B)+1$ competitive ration with the offline optimum. Their algorithm smooths its budget over time, and operates by bidding: $1/(1+\exp(z(t)-1))$ where $z(t)$ is the fraction of budget remaining at time t.

6.8. Experimental Results

In this section we will describe experimental results for the previously described algorithms. We use bids placed through Microsoft's adCenter in two sets of experiments. In the first, we assume that our modeling assumptions from Section 6.1 are correct, and construct a distribution $\mathcal{P}$ from the empirical data for use in simulation. In the second set of experiments, we run the methods on the historical data directly, taken as an individual sequence and thus violating our stochastic assumptions. We will see that in both cases, our suggested algorithm outperforms the other methods discussed. First, however, we discuss an important generalization to the setting that we have considered so far.

6.8.1. Impressions and Clicks

Until now we have assumed that all ad impressions result in a click (i.e. winning an auction results in an automatic click). We will now relax this assumption. Instead, when an advertiser wins an impression, we will suppose that whether a click occurs is an independent Bernoulli random variable with mean r. We call r the *click-through rate*. If a click does indeed occur, the advertiser is charged the market price. Otherwise, the advertiser is informed that an impression has occurred, but maintains its budget.

All the methods described generalize to this setting in a straightforward manner. Nevertheless, it is worth being explicit about how *Greedy Product-Limit* must be modified. First note that the MDP formulation for the problem differs in the definition of the transition probability μ. In particular, we now have $\mu(a,(b,t),(b-\delta,t-1)) = rp(\delta)$ (when $\delta \leq a$), and $\mu(a,(b,t),(b,t-1)) = (1-r\sum_{\delta=1}^{a} p(\delta))$. π_p^* can still be computed using dynamic programming, where:

$$V_p(a, B, T) = (1 - r\sum_{\delta=1}^{a} p(\delta))V_p(B, T-1) + \sum_{\delta=1}^{a} rp(\delta)[1 + V_p(B-\delta, T-1)]$$

and $\pi_p^*(B, T) = \arg\max_{a \leq B} V_p(a, B, T)$

Furthermore, as discussed in Section 6.5, rather than using the Product-Limit estimator, this setting requires that the new algorithm treat doubly-censored data, for which techniques exist Turnbull (1974).

6.8.2. Data

The data used for these experiments were generated by collecting the auction history from advertisers placing bids through Microsoft's adCenter over a six month period. For a given keyword, we let the number of times that keyword generated an auction be its *search volume*, $\texttt{Vol}_k$, and take keyword k to be the keyword with the k-th largest volume.

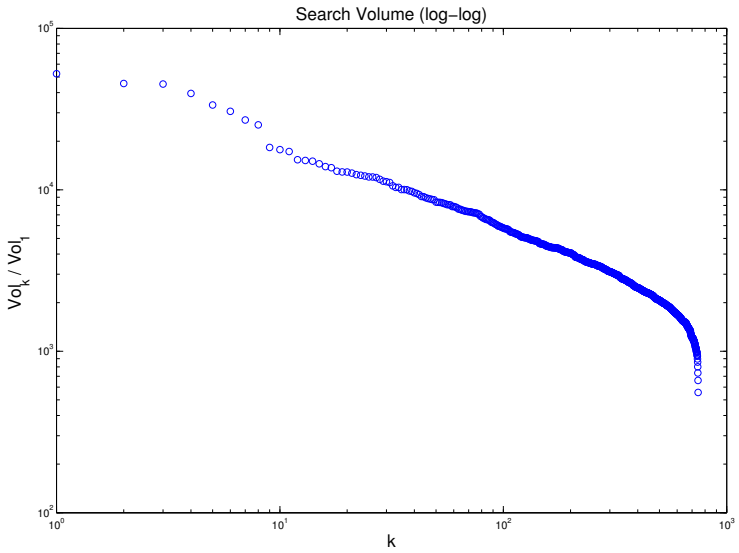

Figure 5: Log-log plot of $\texttt{Vol}_k/\texttt{Vol}_1$ for each k

The distribution of search volume is clearly heavy-tailed (see Figure 5) and is well approximated by a power law over several orders of magnitude.

We ran our experiments on the 100 keywords with largest search volume. As we will discuss further in the next section, the bidding behavior is quite varied among the different keywords

in the data set.

In actual sponsored search-auctions, each bidder is given a *quality-score* for each keyword. Bidders' actual bids are multiplied by these quality scores to determine who wins the auction. For each keyword k, and auction t, let $x_{k,t}$ be the *quality-score-adjusted bid* for the advertiser that historically won the top slot for that auction, and $c_{k,t}$ indicate whether a click occurred.

We take the perspective of a new advertiser with unit quality score. $x_{k,t}$ is the amount that such an advertiser would have needed to bid to have instead taken the top slot (and the amount the advertiser would be charged should they also receive a click).

Finally, we make the single-slot assumption throughout, so even if multiple ads were indeed shown historically, we assume that an algorithm wins an impression if and only if it wins the top slot. In principle, *Greedy Product-Limit* can be modified to operate when multiple ad slots are available. However, we avoid this complication in this work.

6.8.3. Distributional Simulations

The first set of simulations use the historical data $\{x_{k,t}\}$, to construct an empirical distribution p_k on market prices for each keyword k. We also set a fixed click-through rate r_k, for each keyword using the background click-through rate for that keyword (the empirical average of $\{c_{k,t}\}$). While our main results in the next section eliminate the distribution p_k and use the sequence $\{x_{k,t}\}$ directly, we first simulate and investigate the case where our modeling assumptions hold.

Figure 6 demonstrates that the types of distributions generated in this manner are quite varied.

The budget B_k allocated to the advertiser for keyword k is selected so that, optimally, a constant fraction of clicks are available. In other words, the simulations were run with $B_k(T)$ satisfying $V_{p_k}(B_k(T), T) = fT$, where $f = 10\%$. Each experiment was run for 10

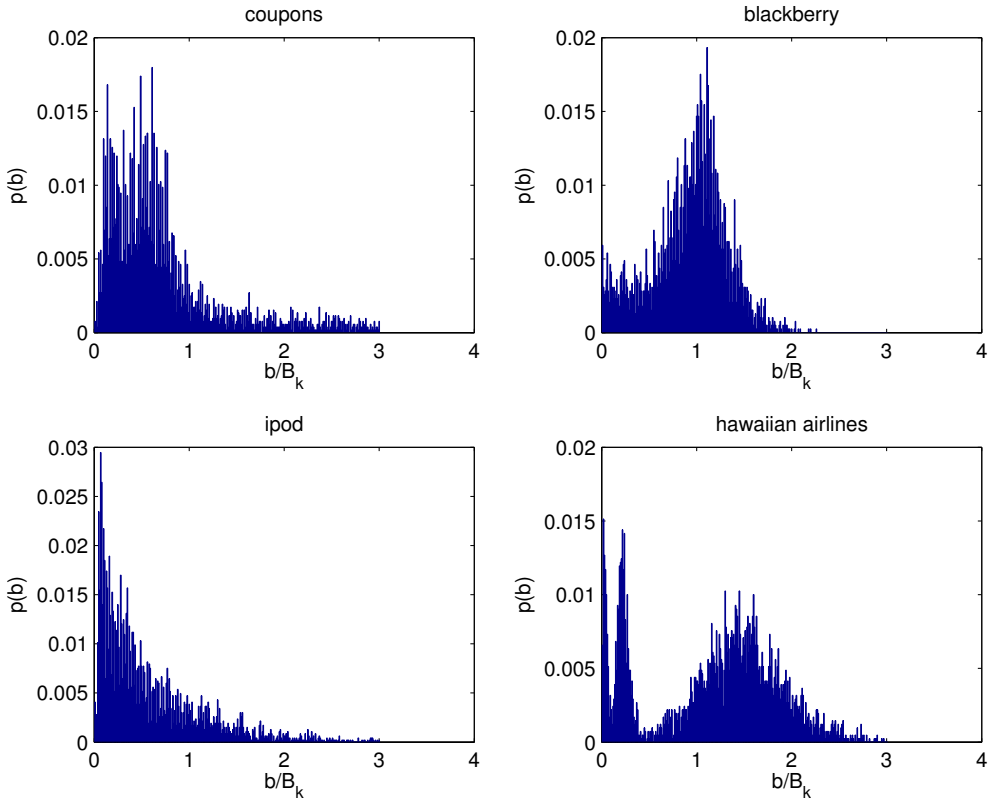

Figure 6: Each plot represents the empirical distribution p_k for a different keyword k. The x-axis represents the bid as a fraction of the total budget B_k allocated to the advertiser for keyword k.

periods containing $T = 100$ auctions each, and 20 experiments were run for each keyword.

A major observation is that *Greedy Product-Limit* converges to the optimum policy on the time-scale of auctions, not periods. This is significant since certain methods considered are doomed to converge on the time-scale of periods instead. For example, each state can be visited at most once by Q-learning in a single period. Furthermore, for a particular time t, the only state corresponding to that t visited is $(B_{u,t}, t)$, (i.e. the state corresponding to the budget held by the algorithm at that time). Similarly, the Fixed-Price algorithm adjusts its bid at the end of every period. Table 1 shows that after just 2 periods, *Greedy Product-Limit* (GPL) has come within five percent of optimal across all keywords. For

each algorithm, the results are the averages of 20 different simulations, and are statistically significant. An unpaired 2-sample t-test between the results for GPL and those of any other algorithm yields p-values that are less than 10^{-20}.

Table 1: Average Competitive Ratio with $V_{p_k}(B,T)$, across all keywords and experiments, after two periods.

Algorithm Name	Competitive Ratio	Std
Greedy Product-Limit	0.9573	0.1704
LuekerLearn	0.8448	0.1842
Fixed-Price Search	0.8352	0.1733
Q-learn	0.7484	0.1786
Budget Smoothing	0.1597	0.2418

6.8.4. Sequential Experiments

The main result of our work comes from experiments run on the real sequential data $\mathbf{d}_k = \{x_{k,t}, c_{k,t}\}_t$. Rather than taking $\{x_{k,t}, c_{k,t}\}$, and constructing the distribution p_k and a click-through rate, as in the previous section, we can use the sequence directly. Each of the previous methods are well-defined if the prices and clicks are generated in this manner, as opposed to being generated by the stochastic assumptions that motivated *Greedy Product-Limit*. We break the sequence into 10 periods of length $T = 100$. In reality, the number of auctions in a period might vary. However, this is minor, as an advertiser can attain good estimates of period-length. We allocate to each algorithm the same budget B_k used in the stochastic experiments.

Recall the notion of offline optimality defined in section 6.7.1. For each keyword k, we can compute the exact auctions that one should win knowing the sequence $\{x_{k,t}, c_{k,t}\}$ *a priori*. We will demonstrate that *Greedy Product-Limit* is competitive with even this strong notion.

We first look at the nature and time-scale of the convergence of *Greedy Product-Limit*.

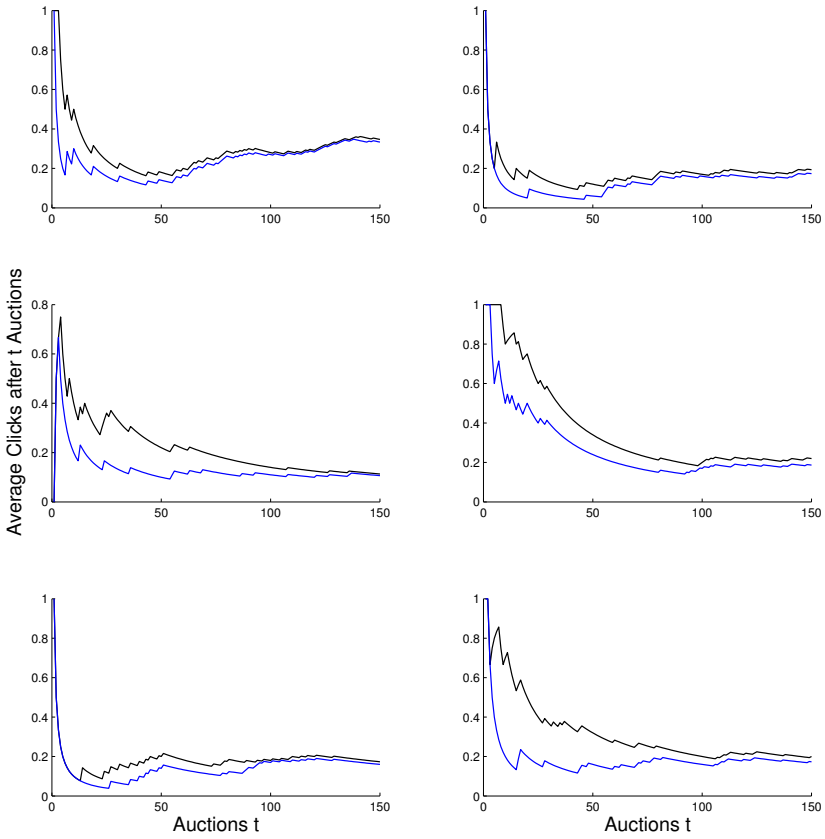

Figure 7: Convergence rates for 6 different keywords. The x-axis denotes auctions t, and the y-axis plots, in black, the number of clicks attained by the offline optimal after t auctions, normalized by t. The blue plot shows the same for *Greedy Product-Limit*. *Greedy Product-Limit* converges in the auction time-scale, not the period time-scale.

We will shortly see that *LuekerLearn*, the modification of *Greedy Product-Limit* attains similar performance. Like *Greedy Product-Limit*, it converges in the time-scale of auctions. However, recalling Figure 4, *LuekerLearn* will sometimes converge to something suboptimal, especially on keywords where there is a lot of variance in the bids. Figure 8 demonstrates this behavior on the sequential data. In fact, let A_k denote the cumulative number of clicks

attained by *Greedy Product-Limit* and L_k denote the cumulative number of clicks attained by *LuekerLearn* after 10 periods for keyword k, defining $Z_k = A_k/L_k$ and $S_k = std(\{x_{k,t}\}_t)$. The observations $\{Z_k\}$ are positively correlated with $\{S_k\}$, with a correlation coefficient of 0.2103 that is significant with a p-value of 0.0357.

Figure 9 demonstrates that, ignoring variance, *Greedy Product-Limit* has indeed converged to optimal across all keywords after just a single period. Let O_k be the offline optimal number of clicks that can be attained for keyword k after 10 periods (the entire data set). Let $A_{k,p}$ denote the cumulative number of clicks attained by an algorithm on keyword k after p periods. For an algorithm that is optimal after a single period, we should expect $A_{k,p}/O_k$ to be roughly $p/10$ for each period p. Indeed, while there are certain keywords for which this doesn't happen, we see that this is true on average.

Figure 10 and Table 2 summarize the performance of the competing methods.

Table 2: Average Competitive Ratio with O_K, across all keywords.

Algorithm Name	Competitive Ratio	Std
Greedy Product-Limit	0.9062	0.1166
LuekerLearn	0.8962	0.1152
Fixed-Price Search	0.6253	0.1395
Q-learn	0.5879	0.1558
Budget Smoothing	0.3105	0.3252

Notice that besides *Greedy Product-Limit*, the only other algorithm that competes with the offline optimal is our modification to the stochastic knapsack algorithm, *LuekerLearn*. As previously discussed, the convergence of *Q-learn* and *Fixed-Price Search* happens on the time-scale or periods, not auctions. We suspect that with more data, both would converge (as they do in the stochastic setting), albeit *Fixed-Price Search* would converge only to the best fixed-price in hindsight. Finally, as demonstrated in Figure 8, when there is large variation in bidder behavior, *LuekerLearn* might stay bounded away from the offline optimal.

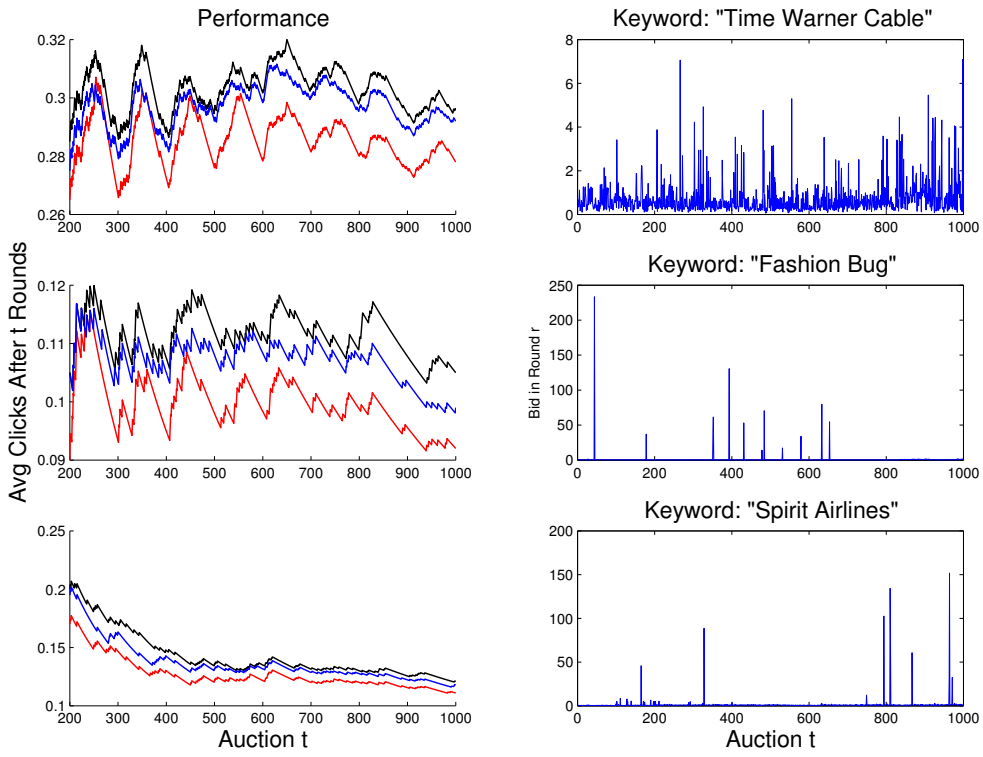

Figure 8: On the left the y-axis plots, in black, the number of clicks attained by the offline policy after t auctions, normalized by t. The blue plot shows the same for *Greedy Product-Limit* and the red for *LuekerLearn*. In all three, *LuekerLearn* is bounded away from the offline optimal. The right side displays $x_{k,t}/\bar{x}_k$ where $\bar{x}_k$ is the average over $x_{k,t}$ for the corresponding keyword. Note the "bursty" nature of the market price, with auctions occurring that set a market price hundreds of times greater than the average.

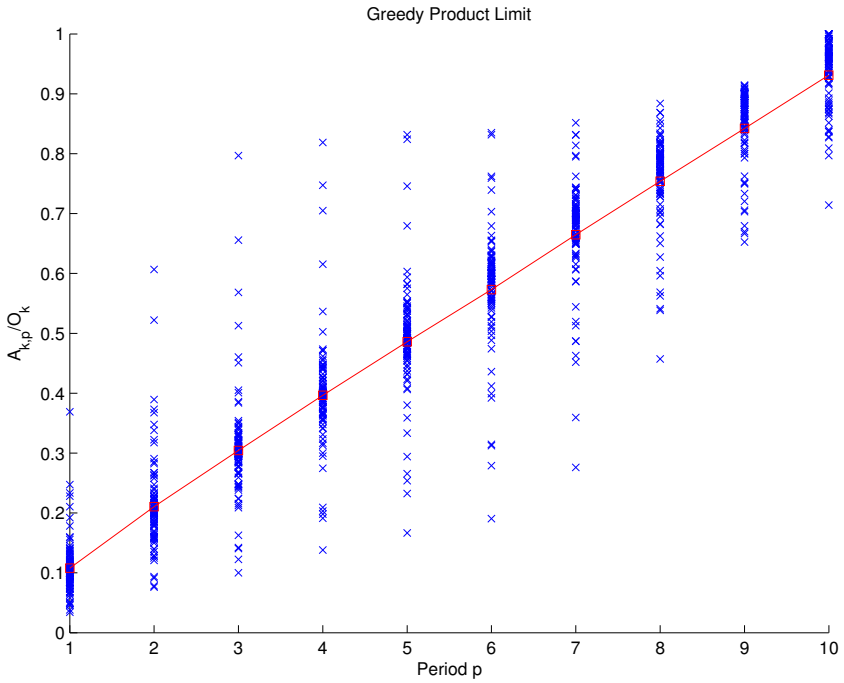

Figure 9: Each scatter point represents $A_{k,p}/O_k$ for a different keyword k, at the end of period p displayed on the x-axis. The line is the mean of $A_{k,p}/O_k$ across all k for a fixed period p.

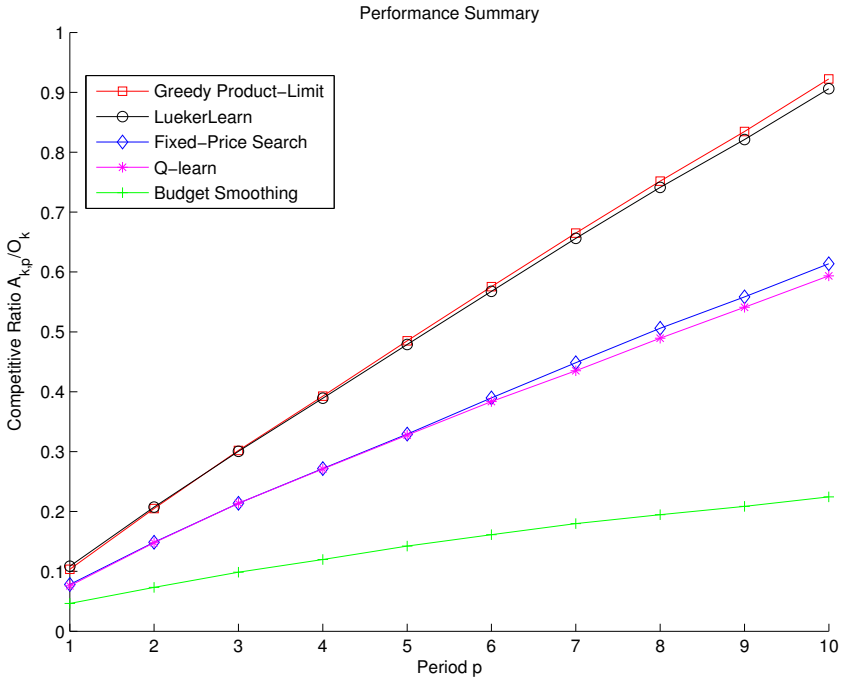

Figure 10: Average of $A_{k,p}/O_k$ across all k for each algorithm.

BIBLIOGRAPHY

N. Abe, A. W. Biermann, and P. M. Long. Reinforcement learning with immediate rewards and linear hypotheses. *Algorithmica*, 37:263–293, 2003.

J. Abernethy, E. Hazan, and A. Rakhlin. Competing in the dark: An efficient algorithm for bandit linear optimization. In *COLT*, pages 263–274, 2008.

J. Abernethy, K. Amin, M. Draief, and M. Kearns. Large-scale bandit problems and kwik learning. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 588–596, 2013.

A. Acquisti and H. R. Varian. Conditioning prices on purchase history. *Marketing Science*, 24(3):367–381, 2005.

R. Agrawal. The continuum-armed bandit problem. *SIAM journal on control and optimization*, 33(6):1926–1951, 1995.

N. Alon, N. Cesa-Bianchi, C. Gentile, and Y. Mansour. From bandits to experts: A tale of domination and independence. In *Advances in Neural Information Processing Systems*, pages 1610–1618, 2013.

K. Amin, M. Kearns, and U. Syed. Bandits, query learning, and the haystack dimension. In *COLT*, pages 87–106, 2011.

K. Amin, M. Kearns, P. Key, and A. Schwaighofer. Budget optimization for sponsored search: Censored learning in mdps. In *Uncertainty in Artificial Intelligence: Proceedings of the Twenty-Eighth Conference*, 2012a.

K. Amin, M. Kearns, and U. Syed. Graphical models for bandit problems. In *Uncertainty in Artificial Intelligence: Proceedings of the Twenty-Seventh Conference*, 2012b.

K. Amin, A. Rostamizadeh, and U. Syed. Learning prices for repeated auctions with strategic buyers. In *Advances in Neural Information Processing Systems*, pages 1169–1177, 2013.

K. Amin, A. Rostamizadeh, and U. Syed. Repeated contextual auctions with strategic buyers. In *Advances in Neural Information Processing Systems*, pages 622–630, 2014.

D. Angluin. Queries and concept learning. *Machine Learning*, 2(4):319–342, 1988.

R. Arora, O. Dekel, and A. Tewari. Online bandit learning against an adaptive adversary: from regret to policy regret. In *ICML*, 2012.

J.-Y. Audibert, S. Bubeck, et al. Best arm identification in multi-armed bandits. *COLT 2010-Proceedings*, 2010.

P. Auer. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3:397–422, 2002.

P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2-3):235–256, 2002.

P. Auer, N. Cesa-Bianchi, Y. Freund, and R. Schapire. The nonstochastic multiarmed bandit problem. *SIAM Journal on Computing*, 32(1):48–77, 2003.

M. Babaioff, Y. Sharma, and A. Slivkins. Characterizing truthful multi-armed bandit mechanisms. In *Proceedings of Conference on Electronic Commerce*, pages 79–88. ACM, 2009.

M. Babaioff, R. D. Kleinberg, and A. Slivkins. Truthful mechanisms with implicit payment computation. In *Proceedings of the Conference on Electronic Commerce*, pages 43–52. ACM, 2010.

Z. Bar-Yossef, K. Hildrum, and F. Wu. Incentive-compatible online auctions for digital goods. In *Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 964–970. Society for Industrial and Applied Mathematics, 2002.

F. Barahona. On the computational complexity of Ising spin glass models. *Journal of Physics A: Mathematical, Nuclear and General*, 15:3241–3253, 1982.

A. Beygelzimer and J. Langford. The offset tree for learning with partial labels. In *KDD*, pages 129–138, 2009.

A. Beygelzimer, J. Langford, L. Li, L. Reyzin, and R. E. Schapire. Contextual bandit algorithms with supervised learning guarantees. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2011.

A. Blum, V. Kumar, A. Rudra, and F. Wu. Online learning in online auctions. In *Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 202–204. Society for Industrial and Applied Mathematics, 2003.

A. L. Blum. Separating distribution-free and mistake-bound learning models over the boolean domain. *SIAM Journal on Computing*, 23(5):990–1000, 1994.

C. Borgs, J. Chayes, N. Immorlica, K. Jain, O. Etesami, and M. Mahdian. Dynamics of bid optimization in online advertisement auctions. In *Proceedings of the 16th international conference on World Wide Web*, pages 531–540. ACM, 2007.

S. Bubeck and N. Cesa-Bianchi. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *arXiv preprint arXiv:1204.5721*, 2012.

S. Bubeck, R. Munos, G. Stoltz, and C. Szepesvári. Online optimization in x-armed bandits. In *Advances in Neural Information Processing Systems 21*, 2008.

M. Cary, A. Das, B. Edelman, I. Giotis, K. Heimerl, A. Karlin, C. Mathieu, and M. Schwarz. Greedy bidding strategies for keyword auctions. In *Proceedings of the 8th ACM Conference on Electronic Commerce*, pages 262–271. ACM, 2007.

N. Cesa-Bianchi and G. Lugosi. Combinatorial bandits. *Journal of Computer and System Sciences*, 78(5):1404–1422, 2012.

N. Cesa-Bianchi, C. Gentile, and F. Orabona. Robust bounds for classification via selective sampling. In *Proceedings of the 28th International Conference on Machine Learning*, 2009.

N. Cesa-Bianchi, C. Gentile, and Y. Mansour. Regret minimization for reserve prices in second-price auctions. In *Proceedings of the Symposium on Discrete Algorithms*. SIAM, 2013.

V. Conitzer and T. Sandholm. Computing the optimal strategy to commit to. In *Proceedings of the 7th ACM conference on Electronic commerce*, pages 82–90. ACM, 2006.

V. Dani, S. M. Kakade, and T. P. Hayes. The price of bandit information for online optimization. In *Advances in Neural Information Processing Systems*, pages 345–352, 2007.

O. Dekel, F. Fischer, and A. D. Procaccia. Incentive compatible regression learning. *Journal of Computer and System Sciences*, 76(8):759–777, 2010.

N. R. Devanur and S. M. Kakade. The price of truthfulness for pay-per-click auctions. In *Proceedings of the Conference on Electronic commerce*, pages 99–106. ACM, 2009.

N. R. Devanur, Y. Peres, and B. Sivan. Perfect bayesian equilibria in repeated sales. 2014.

B. Edelman and M. Ostrovsky. Strategic bidder behavior in sponsored search auctions. *Decision support systems*, 43(1):192–198, 2007.

A. D. Flaxman, A. T. Kalai, and H. B. McMahan. Online convex optimization in the bandit setting: gradient descent without a gradient. In *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 385–394. Society for Industrial and Applied Mathematics, 2005.

D. Fudenberg and J. Tirole. Game theory. *MIT Press Books*, 1, 1991.

D. Fudenberg and J. M. Villas-Boas. Behavior-based price discrimination and customer recognition. *Handbook on economics and information systems*, 1:377–436, 2006.

K. Ganchev, Y. Nevmyvaka, M. Kearns, and J. W. Vaughan. Censored exploration and the dark pool problem. *Communications of the ACM*, 53(5):99–107, 2010.

S. A. Goldman and M. J. Kearns. On the complexity of teaching. *Journal of Computer and System Sciences*, 50(1):20–31, 1995.

O. D. Hart and J. Tirole. Contract renegotiation and coasian dynamics. *The Review of Economic Studies*, 55(4):509–540, 1988.

T. Hegedüs. Generalized teaching dimensions and the query complexity of learning. In *Proceedings of the 8th Annual Conference on Computational Learning Theory*, pages 108–117. ACM, 1995.

M. Jain, J. Tsai, J. Pita, C. Kiekintveld, S. Rathi, M. Tambe, and F. Ordóñez. Software assistants for randomized patrol planning for the lax airport police and the federal air marshal service. *Interfaces*, 40(4):267–290, 2010.

E. Kaplan and P. Meier. Nonparametric estimation from incomplete observations. *Journal of the American Statistical Association*, pages 457–481, 1958.

M. Kearns and U. Vazirani. *An Introduction to Computational Learning Theory*. MIT Press, 1994.

B. Kitts and B. Leblanc. Optimal bidding on keyword auctions. *Electronic Markets*, 14(3): 186–201, 2004.

R. Kleinberg and T. Leighton. The value of knowing a demand curve: Bounds on regret for online posted-price auctions. In *Foundations of Computer Science, 2003. Proceedings. 44th Annual IEEE Symposium on*, pages 594–605. IEEE, 2003.

R. Kleinberg, A. Slivkins, and E. Upfal. Multi-armed bandits in metric spaces. In *Proceedings of the 40th annual ACM symposium on Theory of computing*, pages 681–690. ACM, 2008.

R. D. Kleinberg. Nearly tight bounds for the continuum-armed bandit problem. In *Advances in Neural Information Processing Systems*, pages 697–704, 2004.

D. Koller and N. Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

D. Korzhyk, Z. Yin, C. Kiekintveld, V. Conitzer, and M. Tambe. Stackelberg vs. nash in security games: An extended investigation of interchangeability, equivalence, and uniqueness. *J. Artif. Intell. Res.(JAIR)*, 41:297–327, 2011.

T. L. Lai and H. Robbins. Asymptotically efficient adaptive allocation rules. *Advances in applied mathematics*, 6(1):4–22, 1985.

J. Langford and T. Zhang. The epoch-greedy algorithm for contextual multi-armed bandits. In *Advances in Neural Information Processing Systems 20 (NIPS)*, 2007.

L. Li and M. L. Littman. Reducing reinforcement learning to kwik online regression. *Annals of Mathematics and Artificial Intelligence*, 58(3-4):217–237, 2010.

L. Li, M. Littman, and T. Walsh. Knows what it knows: a framework for self-aware learning. In *Proceedings of the 25th International Conference on Machine Learning (ICML)*, pages 568–575, 2008.

L. Li, W. Chu, J. Langford, and R. E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th International World Wide Web Conference*, pages 661–670. ACM, 2010.

L. Li, M. L. Littman, T. J. Walsh, and A. L. Strehl. Knows what it knows: a framework for self-aware learning. *Machine learning*, 82(3):399–443, 2011.

N. Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine learning*, 2(4):285–318, 1988.

T. Lu, D. Pál, and M. Pál. Contextual multi-armed bandits. In *International Conference on Artificial Intelligence and Statistics*, pages 485–492, 2010.

G. Lueker. Average-case analysis of off-line and on-line knapsack problems. In *Proceedings of the Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 179–188. Society for Industrial and Applied Mathematics, 1995.

S. Mannor and O. Shamir. From bandits to experts: On the value of side-observations. In *Advances in Neural Information Processing Systems*, pages 684–692, 2011.

A. Marchetti-Spaccamela and C. Vercellis. Stochastic on-line knapsack problems. *Mathematical Programming*, 68(1-3):73–104, 1995.

R. Meir, A. D. Procaccia, and J. S. Rosenschein. Strategyproof classification with shared inputs. *Proc. of 21st IJCAI*, pages 220–225, 2009.

A. J. Mersereau, P. Rusmevichientong, and J. N. Tsitsiklis. A structured multiarmed bandit problem and the greedy policy. *IEEE Transactions on Automatic Control*, 54(12):2787–2802, 2009.

M. Mohri and A. Munoz. Optimal regret minimization in posted-price auctions with strategic buyers. In *Advances in Neural Information Processing Systems*, pages 1871–1879, 2014.

R. B. Myerson. Optimal auction design. *Mathematics of operations research*, 6(1):58–73, 1981.

R. Nowak. Noisy generalized binary search. In *Advances in Neural Information Processing Systems 22*, pages 1366–1374, 2009.

S. Pandey, D. Agarwal, D. Chakrabarti, and V. Josifovski. Bandits for taxonomies: A model-based approach. In *SDM*, pages 216–227. SIAM, 2007.

J. Pita, M. Jain, J. Marecki, F. Ordóñez, C. Portway, M. Tambe, C. Western, P. Paruchuri, and S. Kraus. Deployed armor protection: the application of a game theoretic model for security at the los angeles international airport. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems: industrial track*, pages 125–132. International Foundation for Autonomous Agents and Multiagent Systems, 2008.

H. Robbins. Some aspects of the sequential design of experiments. In *Herbert Robbins Selected Papers*, pages 169–177. Springer, 1985.

D. Russo and B. Van Roy. Eluder dimension and the sample complexity of optimistic exploration. In *Advances in Neural Information Processing Systems*, pages 2256–2264, 2013.

A. Sayedi, M. Zadimoghaddam, and A. Blum. Trading off mistakes and don't-know predictions. In *Advances in Neural Information Processing Systems*, pages 2092–2100, 2010.

K. M. Schmidt. Commitment through incomplete information in a simple repeated bargaining game. *Journal of Economic Theory*, 60(1):114–139, 1993.

A. Slivkins. Contextual bandits with similarity information. *The Journal of Machine Learning Research*, 15(1):2533–2568, 2014.

A. Strehl and M. L. Littman. Online linear regression and its application to model-based reinforcement learning. In *Advances in Neural Information Processing Systems 20 (NIPS)*, 2007.

W. Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3–4):285–294, 1933.

L. Tran-Thanh, L. C. Stavrogiannis, V. Naroditskiy, V. Robu, N. R. Jennings, and P. Key. Efficient regret bounds for online bid optimisation in budget-limited sponsored search auctions, 2014.

B. Turnbull. Nonparametric estimation of a survivorship function with doubly censored data. *Journal of the American Statistical Association*, pages 169–173, 1974.

L. G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

V. N. Vapnik and A. Y. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability & Its Applications*, 16(2):264–280, 1971.

H. Varian. Position auctions. *International Journal of Industrial Organization*, 25(6):1163–1178, 2007.

T. J. Walsh, I. Szita, C. Diuk, and M. L. Littman. Exploring compact reinforcement-learning representations with linear regression. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pages 591–598. AUAI Press, 2009.

C. Watkins and P. Dayan. Q-learning. *Machine Learning*, 8(3):279–292, 1992.

Y. Zhou and V. Naroditskiy. Algorithm for stochastic multiple-choice knapsack problem and application to keywords bidding. In *Proceedings of the 17th international conference on World Wide Web*, pages 1175–1176. ACM, 2008.

Y. Zhou, D. Chakrabarty, and R. Lukose. Budget constrained bidding in keyword auctions and online knapsack problems. In *Internet and Network Economics*, pages 566–576. Springer, 2008.