EFFECTIVE ONLINE DECISION-MAKING IN COMPLEX MULTI-AGENT SYSTEMS

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Theodoros Lykouris

August 2019

EFFECTIVE ONLINE DECISION-MAKING IN COMPLEX MULTI-AGENT SYSTEMS

Theodoros Lykouris, Ph.D.

Cornell University 2019

The emergence of online marketplaces has introduced important new dimensions to online decision-making. Classical algorithms developed to guarantee worst-case performance often focus strongly on the worst case; in typical inputs one can perform much better which makes these approaches not practical. Moreover, these marketplaces serve multiple agents who interact in complex ways; this adds important facets to designing online decision-making approaches in these systems. This thesis aims to shed light on both of these issues.

In particular, in the first theme of the thesis, we show how to utilize nice structures in the data to enhance classical worst-case guarantees without requiring that these structures are perfectly present. Instead the performance gracefully degrades as these structures become less present. We discuss how to exploit three such nice structures: existence of a really good alternative, well-behaved randomness, and predictability of future requests.

The second theme of the thesis explores the multi-agent aspect of modern online decision-making which adds important constraints to the classical tasks. In this direction, we discuss pricing under the existence of network externalities (such as ones arising in ridesharing systems), outcomes in evolving game settings with multiple strategic learning agents, and tradeoffs between effective online decision-making and ethical goals regarding non-discrimination.

BIOGRAPHICAL SKETCH

Theodoros (Thodoris) Lykouris was born and grew up in Athens. He completed his undergraduate diploma in the Department of Electrical and Computer Engineering in National Technical University Athens. Subsequently he enrolled as a Ph.D. student to the Department of Computer Science in Cornell University. During his Ph.D. studies, he had the pleasure of doing internships in Microsoft Research India, Google Research New York, and Microsoft Research Redmond. He also spent valuable time as a visiting student at Toyota Technological Institute at Chicago and at the Simons Institute at Berkeley. After graduating from Cornell, he will join Microsoft Research New York as a postdoctoral researcher.

ACKNOWLEDGEMENTS

This thesis would have been significantly poorer without the continuous support of a large number of exceptional individuals. I would like to therefore take the opportunity to thank each of them for helping me mature as a researcher, serving as inspirational forces, providing ample technical and emotional support, and making my graduate years a very pleasant period of my life.

First and foremost, I am deeply grateful to my advisor, Éva Tardos. Beyond her inherent brilliance, her impressive intuition, and her inspiring work ethic, what really distinguishes Éva as an advisor is the extent to which she prioritizes her students' scientific growth and personal well-being. Éva has been extremely generous with her time, actively delving into our research pursuits and offering ample advice regarding both research directions and important career decisions. Her genuine care towards my success is nicely demonstrated by how ubiquitously available she has been to provide feedback regarding even minor issues and how supportive and encouraging she has been towards me building external collaborations. Being mentored by Éva has been an enormous privilege which has undoubtedly shaped my research and life perspective.

At Cornell, I was also fortunate to be surrounded by many other faculty who have had a significant role in my research development. I am very thankful to Jon Kleinberg, Karthik Sridharan, and Sid Banerjee for serving on my committee. Jon greatly facilitated my adjustment in Cornell with his very useful feedback on my undergraduate project (published after I joined) and with his phenomenal teaching style. Since then, he has provided useful suggestions and, in particular, his feedback has significantly improved the structure of this thesis. Karthik has played a pivotal role in familiarizing me with the online learning literature which is a prominent topic in this thesis. Moreover, a question of his in one of my talks eventually led to our first joint paper and, since then, we have spent a lot of hours working together on the whiteboard. Sid also has a profound contribution to my research life. Our joint paper initiated me to the queueing-theoretic literature, boosted my confidence in undertaking projects in new areas, and is an exemplary demonstration of how a class project can lead to a nice research work. Our discussions also helped me form a more holistic view of online decision-making in online marketplaces. Outside of my committee, I would like to thank Bobby Kleinberg and David Shmoys for engaging into intriguing research discussions and relaxing social activities as well as offering invaluable professional advice.

This thesis greatly benefited from multiple internships and research visits throughout my Ph.D. studies; I would like to thank my mentors and collaborators there. In particular, I am grateful to Vahab Mirrokni, Renato Paes Leme, and Sergei Vassilvitskii for helping make my summer at Google Research NYC the most productive period of my Ph.D. This internship offered great research stimulation and enabled me to identify the first theme of this thesis. I am also thankful to Avrim Blum, Suriya Gunasekar and Nati Srebro for hosting me at Toyota Technological Institute at Chicago (TTIC) and initiating me to fairness considerations in maching learning; our joint work figures in the last chapter of this thesis. Thanks to my mentor during my internship in Microsoft Research at Redmond, Nikhil Devanur, who helped me advance my expertise on online algorithms and multi-agent learning dynamics. Finally, I am thankful to the Simons Institute at Berkeley for inviting me to two semester programs on *Economics and computation* and *Real-time decision-making* that I happily attended.

One of the main reasons I enjoyed my Ph.D. experience is undoubtedly my collaboration with other students at Cornell. I had the pleasure to have productive and fun collaborations with Vasilis Syrgkanis, Dylan Foster, Daniel

Freund, and Drishti Wali. Having come through a similar background, Vasilis has been extremely helpful in my adjustment both to Cornell and to research. His attitude towards tackling projects has heavily influenced my research style and I am greatly thankful for his guidance and mentorship. Dylan has been really helpful during my introduction to machine learning research. Daniel has been my main buddy during my graduate studies and our collaboration is probably one of the main reasons for that. Last, working with Drishti has helped me develop mentoring skills; I am very happy with her growth and I am looking forward to see more from her in the near future.

There are many other people that have been important in keeping me happy, motivated, and sane during my graduate studies. I would like to thank all the students in the theory group for creating a very collegial work environment. In particular, thanks to Hedyeh Beyhaghi, Pooya Jalaly, Rad Niazadeh, Daniel Freund, Rahmtin Rotabi, Sam Hopkins, Dylan Foster, Rediet Abebe, Manish Raghavan, Shijin Rajakrishnan, Ayush Sekhari, Drishti Wali, and Makis Arsenis. Special thanks to Sam, Daniel, and Manish for their feedback in preliminary versions of this thesis and, in general, for being amazing friends. I would also like to thank students outside the theory group that made life in Ithaca much more fun; in particular thanks to Elisavet Kozyri who served as my student mentor and took her role really seriously making my adjustment to Ithaca seamless and Steffen Smolka with whom I did most course assignments and shared a roof for a couple of years. Thanks also to Christoforos Mavrogiannis, Christine Diepenbrock, Nirvan Tyagi, and Jonathan di Lorenzo, for making parts of my graduate studies more pleasant. Outside Ithaca, I would like to thank many friends for making my research visits and internships more fun and social. In particular, thanks to Haris Angelidakis, Konstantina Mellou, Katerina Sotiraki, Manolis

Zampetakis, Dimitris Tsipras, Thanasis Lianeas, Sepehr Assadi, Eric Balkanski, Zelda Mariet, Markos Epitropou, Fotis Iliopoulos, and Antonis Papadimitriou.

Finally, my life would have been significantly emptier without the unconditional love and support of a few people. My parents, Nikos and Dina, have done everything they could to make sure that I have the tools and opportunities to discover and follow my passions. Since my early childhood, my father was always challenging me with riddles and mathematical quizes and this probably has an effect on my passion to prove theorems. My mother taught me to express my thoughts, form arguments, and write coherently; a phrase she likes a lot is: *Thought without language is mute; language without thought becomes a cry*. My brother, Yannis, has also been a constant presence in my life. I am thankful to have kept close contacts with friends from my life before starting my Ph.D. who know me long enough to point to things I often neglect; in particular, thanks to Orestis, Marialena, Jenny, Lydia, and Irene for being there for me. Last but not least, I am happy, proud, and excited to share my everyday thoughts and activities with Ariadni, who has been a wonderful life companion and the cornerstone of my happiness and well-being over my graduate years.

The work of this thesis has been supported by grants from the National Science Foundation and and fellowships from Google and Cornell University.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

The question of how to make effective decisions in complex systems lies at the heart of machine learning, algorithmic game theory, and optimization. Machine learning uses past data to develop models for these systems that correctly classify unseen requests. Algorithmic game theory addresses the system inefficiencies caused by individual incentives. Optimization provides techniques to maximize desired objective functions in the resulting models, informing decision-making.

The advent of online marketplaces has added an important dimension to the decision-making process: decisions need to be made in a sequential manner in complex multi-agent systems. For example, Google and Facebook need to sequentially decide regarding the selection and pricing for millions of ads every second, relying on effective advertising as their main business model. Uber and Lyft transform transportation in urban centers via dispatch decisions that change the configuration of their systems, thereby affecting future customers.

This thesis offers a principled approach towards effective and responsible online decision-making in such multi-agent systems, centering around two themes. The first exploits nice data patterns in a robust way, while the second addresses complications in multi-agent online decision-making. Concretely we aim to:

Theme I: ***Provably enhance*** *online decision-making if* ***data have a nice structure****, being* ***robust to this structure not being perfectly present****.*

Theme II: *Address* ***economic and societal*** *issues with online decision-making in modern applications where* ***multiple selfish agents*** *interact.*

1.1 Overview of results

In order to achieve provable guarantees in online decision-making, there are two classical approaches. The more statistical approach involves some strong assumption on the data (for example, that they are identically and independently distributed across time). An alternative viewpoint is to make no assumptions about the underlying distributions and aim for worst-case guarantees that are robust against even adversarial inputs.

The first theme of this thesis (Section 1.2 and Chapters 2-4) bridges the aforementioned two approaches in online decision-making. We show how to utilize nice structures in data, arising in modern applications, to enhance performance guarantees while retaining good performance when these structures are not perfectly present. In particular, the guarantees we provide smoothly deteriorate as the corresponding structures become less and less present. Examples of nice structures in the data we utilize to enhance online decision-making are:

- The existence of an option that is almost perfect; the guarantees degrade with the loss associated to this best option (Section 1.2.1 and Chapter 2).
- Well-behaved randomness in the performance of available options; bounds decay with how far from independently and identically distributed (i.i.d.) data end up being (Section 1.2.2 and Chapter 3).
- Predictability of future requests; the guarantees degrade with the eventual inaccuracy of these predictions (Section 1.2.3 and Chapter 4).

Subsequently, we move forward to discuss challenges that arise in online decision-making due to the multi-agent nature of modern online markets. Understanding the effect of multi-agent interactions in offline settings is widely studied within operations research, economics, and social sciences. However, classical optimization approaches diregard the online aspect of these decisions and the future externalities they cause to different agents in the resulting ecosystems. More recently, there has been attention towards incorporating these issues [106, 63] but there are important challenges that remain uncharted.

The second theme of this thesis (Section 1.3 and Chapters 5-7) addresses challenges in settings where multi-party interests overlap in a robust and responsible manner. We tackle three particular instantiations of such challenges:

- Pricing decisions in ridesharing often affect future requests in different locations, traditionally treated as independent (Section 1.3.1 and Chapter 5).
- Platforms are not the only data-savvy entities; advertisers can also use data to adapt their behavior towards enhancing their individual utility, potentially causing system ineffeciencies (Section 1.3.2 and Chapter 6).
- Optimization goals may conflict with ethical concerns such as discrimination against minority groups; this conflict is exacerbated when decisions are made online and data are not i.i.d. (Section 1.3.3 and Chapter 7).

Below we expand on all these issues. We briefly illustrate the main questions targeted, the main results obtained, and the technical highlights from each work.

1.2 Theme I: Exploiting nice data properties in a robust way

In the first theme of this thesis, we show how to enhance online decision-making via utilizing nice structures that data exhibit in a way robust to these structures not being perfectly present. When all data are available offline, the decision-maker can perform data analysis, identify desired nice structures, and apply optimization techniques that actively use them. The online aspect of many applications makes this task significantly more difficult for multiple reasons. First, once a structure is eventually identified, the decision-maker may have already made many suboptimal decisions, irrevocably harming the system's performance. More critically, since not all data are available in advance, the decision-maker may mistakenly identify a non-existing structure early in the process. This may also arise due to self-interested entities directing the decision-maker towards such mistakes and can have severe implications on future performance if the employed optimization techniques are not robust to these mistakes.

To illustrate the complications that arise while making online decisions in complex systems, let's introduce a few particular tasks which we use as running examples. A routing application such as Google Maps or Waze wishes to suggest to their users routes minimizing their travel time. An online advertising platform such as Google or Bing Ads wants to identify the most profitable ad to display in response to a search query. A health care provider aims to offer the most effective treatment to a patient with particular symptoms. In all of these scenaria, the platform tries to extract which option is the most effective in order to use it. Since this information is not known, the platform needs a way to learn it.

The classical machine learning approach (batch learning) is not very suitable to inform decision-making in the aforementioned tasks. Batch learning consists of collecting enough samples and subsequently using them to learn models that perform well on unseen data. This approach works well when data come from i.i.d. distributions and we can obtain access to clean samples from these distributions, but can be very unreliable in dynamic environments with multiple self-interested agents such as the tasks we described above. For instance, in routing, the travel times of different routes can be significantly affected by temporal and not easily predictable events (an accident in a street causing high congestion, the end of a baseball game affecting the traffic patterns around the stadium). Moreover, travel times are also affected by the decisions of all the drivers in the system who have their own individual goals and assuming that their behavior is completely unchanging across time is unrealistic.

One approach towards circumventing some of the aforementioned pitfalls is the literature of online learning that aims to adaptively balance the need of exploration with exploiting options that have been effective in the past. Initiated by the seminal works of Robbins [113], Blackwell [25], Hannan [64], and Gittins [58], the online learning framework formalizes this explore-exploit trade-off under various modeling assumptions regarding the underlying system. Despite cleanly capturing the essence of this trade-off, these classical approaches do not provide very meaningful guidance for the applications we discussed as they suffer from various issues. They often assume access to an unrealistic amount of information in the form of knowing accurate priors on the performance of different actions such as Markovian bandits [58, 126], or in the form of receiving feedback even for options not selected such as the experts setting [25, 64, 54, 72]. Alternative approaches that are prior-free and not full-information such as stochastic bandits [113, 86, 11] suffer from an orthogonal issue of relying on the performance of all options being completely i.i.d. across time. Finally, classical works that avoid these assumptions such as adversarial bandits [12] tend to employ a completely worst-case approach and revert to overly cautious exploration to keep an up-to-date view of the changing world, resulting in ineffective guarantees.

In Sections 1.2.1-1.2.2 (and Chapters 2-3), we focus on ways to make online learning more practical by avoiding shortcomings of classical approaches. In Section 1.2.3 (and Chapter 4), we extend our scope to the notion of competitive analysis that also captures the notion of state in the underlying systems. The general philosophy behind our contributions is to design algorithms with data-dependent guarantees that improve on the worst-case when data exhibit a nice structure, but do not rely on such a structure being perfectly present and gracefully degrade as the input becomes less well-behaved.

1.2.1 Contribution I: Mitigating exploration in online learning

To present the first contribution of this thesis, we need to provide a slightly more formal description of non-stochastic online learning. We have a set of alternatives (e.g. the paths in the routing example) which are typically referred to as *arms*. At every round, the learner (e.g. Google Maps or Waze) selects an arm, possibly in a randomized way. Then losses are selected for each of the alternatives; since we do not want to make any i.i.d. assumption in the particular application, the losses are assumed to be selected adversarially and are only assumed to be bounded in $[0, 1]$. The only information that the adversary does not know is which arm was selected (if the algorithm is randomized, the adversary only knows the probability that each arm is selected). In most applications, there is no explicit adversary but this stronger framework enables us to capture the non-stochasticity of the environment. The learner incurs the loss of the selected action and observes some feedback, e.g. the losses of all arms regardless whether they were selected (full information), only the loss of the selected arm (bandit feedback), or some partial feedback in between (as we discuss below).

The learner wishes to minimize the aggregate loss of her selected arms over time (the average travel time that a user experiences). However, this quantity on its own does not provide meaningful guidance of whether the learner uses an effective algorithm: the average travel time may be high because the algorithm is making suboptimal decisions but it may also be high because there is no route that has a small travel time. As a result, to evaluate the performance of the algorithm, online learning literature compares the loss of the algorithm to the loss of a benchmark. The classical notion of regret uses as comparator benchmark the fixed action that is best in hindsight (has the smallest ex post aggregate loss). Typical regret bounds are sublinear in the time horizon, i.e., the average regret goes to 0 as time goes by. These *no-regret guarantees* capture the idea that, if there exists a consistently good action, the algorithm should at some point realize it and follow that action – the learner should not look back at the end of time and regret not having followed that action. Note that, despite this property, the regret may still be the dominant term when there exists a really good action.

Main question. An important issue that arises when employing online learning while only receiving partial feedback is the need of over-exploration. The natural tendency to deal with partial feedback is to explore often all arms, including suboptimal ones, to ensure an up-to-date view on how well all actions behave. This results in selecting suboptimal arms often which leads to ineffective regret guarantees and is a big roadblock towards widely employing online learning.

One approach to deal with this problem is to aim for guarantees that, if satisfied, prohibit this over-exploration. One such category is the *small-loss regret guarantees*, which require the algorithm to achieve a regret that is sublinear in the loss of the best arm. Note that when the best arm has aggregate loss close to 0, the algorithm is not allowed to select suboptimal actions often. Algorithms that over-explore to keep an up-to-date view on all arms cannot satisfy this guarantee which means that achieving small-loss guarantees, to a large extent, limits this excessive exploration. Small-loss guarantees are also a particular way to exploit a nice structure in data (best arm being almost perfect) while gracefully degrading as we deviate from it: regret scales with the loss of that arm.

Small-loss guarantees are particularly challenging when one receives partial feedback, c.f. [3]. With full information, these bounds are easily achieved by most online learning algorithms such as multiplicative weights [54] or follow the perturbed leader [72]. This happens because, when the learner receives full information, she automatically obtains an up-to-date view about how all actions are behaving without selecting them. On the other hand, for partial feedback, the landscape is significantly less clear. Outside of the work we present here, there are only a few such guarantees that focus on restricted feedback settings: label-efficient prediction [39], pure bandits [5, 53], combinatorial semi-bandits [104], and (subsequently to our work) contextual bandits [33]. All of them rely on algorithms tailored to the setting and give guarantees that only hold in expectation for the weaker notion of pseudoregret that compares to an arm fixed in advance (not the best in hindsight). As a result, we ask the natural question:

What is a general recipe to derive small-loss bounds with partial feedback ?

Result. To approach this question, we focus on a general combinatorial feedback setting, the graph-based feedback introduced by Mannor and Shamir [96]. Before selecting an arm, the learner observes a time-varying undirected graph determining the feedback structure. In particular, the learner observes the loss of the selected arm but she also observes the loss of all neighboring arms (and therefore has access to extra information). This model captures full information (complete graph) and bandit learning (empty graph) as two extremes. Interestingly, it also captures other important partial feedback settings such as contextual bandits [12, 87, 2, 33] and, with a small modification, combinatorial semi-bandits [72, 10, 104]. Alon et al. [6] provided regret bounds for graph-based feedback that scale with the independence number of the graph (the appropriate feedback dimension in the setting); however their guarantees also scale with the time horizon and suffer from the over-exploration issue we discussed before.

In joint work with Karthik Sridharan and Éva Tardos [93], we provide a general way to obtain small-loss regret guarantees for the graph-based feedback setting. Our algorithm takes as input a full information algorithm with a small-loss guarantee (these algorithms are ubiquitous in the litarature), and seamlessly transforms it, in a black-box way, to an algorithm with a small-loss guarantee for the graph-based feedback. Our guarantee holds with high-probability and scales with the maximum independence number of the graphs. In a black-box way, the dependence on the loss $L^\star$ of the best action is $(L^\star)^{2/3}$ but, for particular settings, we use specific algorithms to derive an optimal $\sqrt{L^\star}$ guarantee. Interestingly, even for the special case of bandits, our results are the first to provide this guarantee with high probability. We elaborate on these results in Chapter 2.

Technical highlight. The crux of this general reduction is to ignore low-performing actions but treat them optimistically, allowing them to recover. This requires some background. The classical strategy to deal with partial feedback is to reduce it to full information. However, a full information algorithm expects to receive the losses of all the actions, which is not available in the partial feedback setting. For this reason, classical approaches create an estimator of the losses, ensuring that the estimated losses behave as if they were the actual losses (i.e. the estimator is unbiased and therefore the expected estimated loss is equal to the actual loss for any arm). In order to have good regret guarantees, partial-feedback algorithms try to create such an estimator while not pulling suboptimal arms too often. However, if an arm is not observed often, the variance of an unbiased estimator for its loss will be relatively high. Since this variance ends up in the eventual regret bound, the algorithms need a way to control it by obtaining more information about these arms. Classical algorithms such as EXP3 [12] and EXP3-DOM [6] achieve this via mixing the action distribution with a uniform distribution, ensuring that each action is selected (and therefore also observed) with a big enough probability. However, this means that at every round they select suboptimal arms with big enough probability which results in regret bounds scaling with the loss of the worst arm (rather than the best).

We follow an alternative approach by temporarily freezing (not selecting) the low-performing arms. This resolves the variance problem but creates another issue: estimated losses no longer accurately capture the actual losses for the frozen arms (they may be observed with probability 0). It turns out that, to resolve this, it suffices to credit these arms optimistically treating them as perfect while we ignore them. The idea of freezing arms towards small-loss guarantees was initially suggested by Allenberg et al. [5] who derived pseudoregret guarantees for pure bandits. We extend this technique by making it black-box, high-probability, and dealing with the more involved graph-based feedback setting. This setting poses the extra complication that, when an arm gets frozen, neighboring arms may lose probability of observation and may need to subsequently also get frozen. A nice technical contribution is that we control this snowball effect via a *double-thresholding* technique based on a potential function analysis.

1.2.2 Contribution II: Robustness to adversarial corruptions

Although the environment in online markets is rarely completely i.i.d., there are important applications where data is mostly i.i.d. As an example, consider online advertising: when a user arrives in a particular website, a platform such as Google Ads needs to decide which ad to display. For a particular type of user, it is important for the platform to display the ad that will be more likely to get clicked; this ensures that the user receives relevant content and also provides revenue to the platform which is typically paid per-click. Each ad is associated with the so called click-through rate, which is the probability that, if displayed, it will get clicked. The platform does not know this quantity and needs to explore different alternatives to understand which is the most profitable. This is a canonical example of stochastic bandit learning where the reward for each of the alternatives comes from i.i.d. distributions. Recommender systems exhibit similar issues; the alternatives there correspond to restaurants that, for example, Yelp needs to recommend in a particular area and the stochasticity relates to the quality of experience of a typical user as evaluated by the number of stars.

Stochastic bandit learning exploits the fact that, when the input is i.i.d., the alternative with the highest mean can be learned and subsequently be repeatedly selected to optimize the performance. This task of learning the most profitable alternative is easier when the mean of the best arm $a^{\star}$ is significantly better that the means of other arms; the difference between the mean of $a^{\star}$ and of another arm a is typically called the *gap of the arm a*. The improved guarantees for the stochastic case depend only logarithmically on the time horizon and also scale with the inverse of these gaps which captures how easily identifiable the best arm is. Classical algorithms like Upper Confidence Bound (UCB) [11] and Active Arm Elimination [51] retain empirical means for each arm (based on the average experienced reward) and confidence intervals around them (helping them position the actual means of the arms). This enables them to cease selecting arms that are very unlikely to be the best. Other algorithms such as Thompson sampling [124] employ a randomized way to capture these empirical means but they also achieve gap-based guarantees [4].

Main question. The second contribution of this thesis is to address an important limitation of these classical stochastic bandit learning approaches: in reality, data is not completely stochastic and is often corrupted by self-interested adversarial entities. In online advertising, there is the phenomenon of click fraud where a competing advertiser may try to harm the most profitable alternative $a^\star$, to increase her own displays. One instantiation of click fraud is that the competing advertiser creates bots that obtain fake impressions and, when $a^\star$ is displayed, deliberately not click the ad, misleading the platform to conclude that $a^\star$ is not a profitable ad and therefore it should not be displayed often. Similar attacks can arise in recommender systems with paid fake reviews.

Companies try to detect and mitigate this fraudulent activity, but we cannot hope that it is completely eliminated. Google spends a lot of resources to try to identify activity coming from bots and correct for that. Yelp requests users to report offers for paid fake reviews in order to punish the restaurants that deploy such strategies. Therefore we can expect that most of the attempted fraudulent activity is stopped. However, there is no hope that all of the activity will get caught and some corruption will end up in the data.

The problem is that classical methods fail to be robust to even little fraudulent activity if they utilize the inherent stochasticity in the rewards. Stochastic bandit approaches are easily fooled even by small amounts of corruption and often eliminate (or mostly ignore) the most profitable arm $a^\star$. The other extreme of adversarial bandits which we elaborated in the previous subsection is not fooled by the corruptions but also does not exploit the fact that most of the input is stochastic. Prior to our work, the best approach towards the problem is the literature on best of both worlds [34, 120, 14, 119] which designs algorithms that simultaneously achieve the stochastic guarantee if the input is i.i.d. while also retaining worst-case guarantees. This line of work does not handle the typical case where data are not completely i.i.d. but there is only a minimal amount of corruption in the data. Addressing this limitation, we ask the following question:

Can we make stochastic bandits robust to small amounts of corrupted data?

Result. To tackle this question, in joint work with Vahab Mirrokni and Renato Paes Leme [92], we introduce a model that slightly modifies the stochastic bandit learning framework to incorporate corruptions in the data. More precisely, each arm is associated with a distribution that is fixed across time – this is the classical stochastic bandit learning assumption. At every round, rewards are drawn from this distribution and, at the same time, the learner commits to a probability distribution across the set of k arms. However, unlike stochastic bandit learning, an adversary subsequently corrupts the feedback that the learner observes and returns as feedback some corrupted value in $[0, 1]$ instead of the actual realized reward. If the adversary never changes the feedback then we are in the purely stochastic setting; if she changes it every single time then we are in a heavily corrupted setting where one cannot hope to exploit the stochasticity in the data. Our goal is to robustify the design of stochastic bandit learning algorithms so that they can accommodate a modest amount of corruption in the data without knowing in advance how much this amount is.

In that direction, we provide an algorithm that achieves this desired robustness. Our guarantees have three very nice properties. First, we obtain (up to a logarithm) the gap-based guarantee of classical stochastic bandits when there is no corruption in the data; as a result, the extra penalty that we pay to achieve this robustness is relatively small. Second, our guarantees degrade gracefully with the amount of corruption in the data; in particular, the decay in performance is linear with the total corruption that the adversary injected in the data. We note that this linear degradation is unavoidable even in simple instances. Third, our guarantees are agnostic to the amount of corruption: we do not need to know how much corruption occurs in the data. This is very important in the applications we discussed as, if our algorithm is tailored to a particular level of corruption, it may aim for a pessimistic bound that will ruin the stochastic guarantee when there is close to no corruption. More importantly, if the algorithm has a hard-coded level of robustness, it is easily gameable by an adversarial entity that just needs to add a little more corruption. Finally, our guarantee holds with high probability instead of weaker notions of expected performance; this helps to mitigate the effect of such attacks. We elaborate on these results in Chapter 3.

Technical highlight. Our algorithm is based on a multi-layer random sparsification technique that extends the Active Arm Elimination stochastic bandit algorithm. Active Arm Elimination selects arms in a round-robin fashion until their empirical means concentrate enough so that the difference in empirical means gives confidence that the dominated arm is suboptimal; at this point, it eliminates the dominated arm. This typically takes a logarithmic number of rounds as then concentration bounds kick in. An adversary can mislead this algorithm by corrupting the feedback of the optimal arm for the initial logarithmic rounds, leading the algorithm to eliminate the optimal arm and therefore make mistakes in the remainder of time.

To robustify this algorithm, we run parallel versions of Active Arm Elimination (layers) and, at each round, randomly select a layer with decreasing probabilities. Our approach applies broadly against any adversary but, to obtain intuition, consider the adversary who corrupts just the initial rounds. Layers selected with smaller probhability receive only a few corrupted samples and keep exploring even when the adversary stops corrupting. As a result, the majority of their data are not corrupted and they are not fooled by the adversary. Our technique seamlessly combines the layers ensuring that the first robust layer corrects the mistakes of all less robust layers. Crucially, we never need to identify this robust layer which makes our algorithm agnostic to the level of corruption.

1.2.3 Contribution III: Online algorithms with predictions

Although online learning provides a clean framework to reason about online decision-making, it ignores important externalities among decisions present in most modern systems. In online learning, the decisions are, to a large extent, decoupled and are only connected via the information learned regarding the system. This enables addressing the explore-exploit trade-off which is an important consideration in online decision-making. However, in most systems, current decisions also affect the state of the system and alter the available options. For instance, in two-sided markets such as TaskRabbit, matching customers to a service provider may make the latter unavailable for future requests possibly more amenable to their skill set. Similarly, pricing decisions with limited supply may have externalities to future customers, affecting the product's future availability.

Competitive analysis can be thought as the analogue of adversarial online learning for settings where state is an important consideration. Compared to online learning, competitive analysis results tend to compare to a stronger benchmark (the optimal online algorithm instead of the optimal action in hindsight). On the other hand, the guarantees are weaker (multiplicative instead of additive).

Despite their fundamental theoretical contributions, works in competitive analysis suffer from not being very practical, an issue we already discussed with respect to classical adversarial online learning. Over the last couple decades, competitive analysis has addressed many important settings where state is an issue, such as bipartite matching [75, 47], paging [52, 18], and *k*-server [83, 32]. These works offer valuable paradigms that enhance our understandning of powerful techniques such as the online primal-dual analysis [36] or online mirror descent [32]. However, since competitive analysis wants to be robust against the worst-case, the resulting algorithms suffer again from a need to be conservative and do not tend to exploit the fact that data may enjoy a nice structure.

Main question. One particular such nice structure that the current data-driven era arms us with is that the future is often predictable in a relatively accurate manner. This is enhanced by the rise of machine learning heuristics based on deep learning and data modeling analyses. However, these techniques tend to not have robust guarantees and are prone to errors, for instance, due to outliers or adversarial examples. As a result, the empirical success of machine learning and the robust techniques of competitive analysis pose the natural question:

How can we take advantage of the predictive power of machine learning without sacrificing the worst-case robustness of competitive analysis?

To understand this trade-off, let's consider the paging problem. In the classical (unweighted) paging, there is a cache of size k that can be used to serve future requests fast, and requests arrive sequentially. If the element requested is in the cache, this corresponds to a cache hit and the element is served at zero cost. Otherwise, we have a page fault or cache miss and we need to wait to bring the element in the cache. We therefore incur a cost (in the unweighted case, a cost of 1) and we need to also decide which element to evict from the cache to load the requested element. The classical application of caching is in computer systems where the cache corresponds to physical memory. More recently, the setting has found important applications in storing, say, Youtube videos for companies such Akamai, or saving pages in the cloud for companies such as Microsoft.

So how can one approach this problem? If the future sequence can be perfectly predicted, the simple greedy Bélády algorithm [22], that evicts elements arriving further in the future, performs optimally. At the absence of this hindsight, the competitive analysis approach is settled. Almost any reasonable deterministic scheme such as First In First Out (FIFO) or Least Recently Used (LRU) achieves a competitive ratio of $\Theta(k)$. Surprisingly, reverting to randomized schemes leads to an exponential improvement in performance of $\Theta(\log(k))$ [52]. On the heuristics side, the computer systems community has developed multiple smart data-mining schemes to better exploit properties such as locality of reference.

Result. To combine the predictive power of these heuristics with the worst-case robustness of competitive analysis, in joint work with Sergei Vassilvitskii [95], we incorporate machine learned predictions to the caching task. In particular, we assume that when an element arrives, we get a prediction about the next time it will arrive again in the future. We do not make any assumption on the nature of the predictor and therefore this information may be erroneous; the hope is that it may also often contain useful predictions. To quantify the error of the predictor, we can use many metrics; here we focus on the ℓ_1 error η of the predictor: total absolute distance between predicted arrivals and actual arrivals for all elements.

We aim for three important desiderata, which are essential to appropriately combine predictions and competitive analysis. First, we want almost perfect performance when the predictor is perfect (consistency). Second, since the predictions will not be perfect, we want graceful degradation in performance with the error in the prediction (robsutness); ideally in an optimal rate. Finally, regarldess of how good the predictor is, we want to have performance comparable to the one of the best online algorithm (worst-case competitiveness).

Our algorithm achieves these desired properties and has multiple other practical features. Regarding the bound, our algorithm achieves a competitive ratio of $2 \cdot \min\Big(1 + \sqrt{\eta/\textsc{Opt}}, 2\log(k)\Big)$ where $\textsc{Opt}$ refers to the optimal number of cache misses in the ex-post sequence. This is a factor of 2 worse than both the optimal offline algorithm (if predictions are perfect, i.e. $\eta = 0$) as well as the classical online algorithm Marker that is $2\log(k)$-competitive. At the cost of this extra factor, it allows us to seamlessly interpolate between perfect and completely inaccurate predictions without knowing in which regime it lies in. Beyond the particular bound, our approach can provide a more robust version of the Least Recently Used (LRU) algorithm. Despite its practical empirical performance, LRU suffers from competitive ratio of k (exponentially worse than the guarantees of the best randomized algorithms). Through the lens of our framework, we can take advantage of the predictive power of LRU while at the same time capping its worst-case performance by $\Theta(\log(k))$. We elaborate on these results in Chapter 4.

Technical highlight. Our algorithm, which we term *Predictive Marker*, is a slight predictor-based modification of the classical Marker online algorithm. Marker works in phases; at the beginning of the phase all elements in the cache are unmarked and when an element comes, it gets marked. At the event of a cache miss, it never evicts a marked element (to ensure that it evicts elements that have not arrived very recently) but instead it evicts an element among the unmarked elements uniformly at random, which leads to the logarithmic competitive ratio. We only alter the tie-breaking rule across the elements that are unmarked: instead of evicting unmarked element uniformly at random, we do that according to the predictions. To achieve the desired trade-off, we keep a *blame graph* which enables us to control the error of the predictor with respect to the optimal solution. When the predictor is locally inaccurate, we locally switch to random evictions among unmarked elements to guarantee a worst-case competitive ratio.

1.3 Theme II: Multi-agent online decision-making

In the second theme of the thesis, we broaden our viewpoint to discuss settings where the multi-agent nature of modern markets adds an important novel dimension to online decision-making. In the first theme, we assumed that the system designer is able to enforce any outcome (a route in Waze, an ad to display in Google Ads, or a page to evict in caching). The loss of different outcomes and the feedback observed was affected by the multiple different parties in the system, but we assumed that the designer can enforce the desired outcome. For example, in routing, although the travel time experienced was affected by the decisions of other agents and the resulting congestion they caused, the user could not deviate from the prescribed strategy and necessarily followed the suggested path.

In modern two-sided markets, platforms may often need to think ahead about the fact that its selected actions are implemented by and on multiple different agents. An agent may not follow the recommended suggestions if these suggestions do not align with her incentives. In fact, having access to past data, she can also employ online learning techniques to find strategies better serving her own goals, misreport her true valuations, or even abstain if this better aligns with her individual objectives. Moreover, algorithmic decisions directly affect the experience or opportunities of different people. As a result, optimization methods need to also be thoughtful about societal concerns such as privacy or fairness, potentially sacrificing effectiveness to avoid compromising such issues.

The second theme of this thesis aims to improve our understanding of how strategic behavior and societal issues in multi-agent decision-making affect systems where the decisions are made online. Despite recent focus on multi-agent questions, most works disregard the online aspect of this decision-making which often introduces novel challenges. In Section 1.3.1 (and Chapter 5), we study pricing, which is maybe the most basic representative of optimization under strategic behavior, in the ridesharing context that introduces multiple complex spatial externalities. In Section 1.3.2 (and Chapter 6), we examine whether the fact that agents can also employ online learning techniques to adapt their behavior in dynamic environments introduces further inefficiencies in the underlying systems. Finally, in Section 1.3.3 (and Chapter 7), we discuss whether the online learning techniques we previously discussed are compatible with different group fairness notions and whether there are inherent trade-offs between effectiveness and group fairness in such online decision-making settings.

1.3.1 Contribution IV: Dynamic pricing in ridesharing

Pricing is arguably the most basic setting where the system designer needs to take the incentives of the agents into consideration. From sports events to airline tickets, pricing is the simplest revenue management technique and therefore lies at the heart of many works in economics, operations research, and theoretical computer science. The simplest online pricing setting is the so called *prophet inequalities* [85] where the different agents i arrive online and have values drawn from distributions $\mathcal{F}_i$ known to the designer. The designer wants to set prices in an online manner aiming to maximize her revenue knowing that the agents are *price-taker*, i.e. they will only purchase the good if their value is above the price. This problem can be formulated as a Markov Decision Process where the goal is to find the desired stopping time but there are also simple threshold-based schemes with a single threshold that achieve constant approximation ratios.

Main question. The rise of online markets has significantly complicated the complexity of these online pricing decisions; one of the best examples to illustrate this is a ridesharing application such as Lyft or Uber. In traditional pricing, there is a straightforward relation between the price displayed to a user and the availability of the good in the future: if the price is higher, the user is less likely to purchase the good and therefore the good is more likely to be available for future users. In ridesharing, we tend to have users in many different locations and the *good* is reusable as it corresponds to a driver providing a ride to the customer and this driver can be useful for future customers as well. As a result, a lower price at a location means that the driver is less likely to stay there to serve future local requests, but may help the driver serve another possibly profitable request in the destination of the customer – this can propagate throughout the system (affecting its state). These *complex network state externalities* of any single pricing decision makes this setting significantly more complicated than traditional pricing. Tackling these complexities, we pose the following question:

Can we design effective pricing at the face of network state externalities?

Result. To study this question, in joint work with Siddhartha Banerjee and Daniel Freund [17], we focus on a queueing-theoretic modeling of the setting as prominent in the literature of shared vehicle systems. In our model, we have n discrete locations (nodes) that correspond to the discretizations that such ridesharing companies employ in all their decisions; we also assume that there are m drivers (units). To isolate the first-order effect that we wish to study, we assume that the number of drivers is fixed and that the drivers are not strategic. For any pairs of nodes (i, j), there is a demand of price-taker customers that want to get rides; we assume a continuous-time (Poisson) arrival model and fixed value distributions $\mathcal{F}_{ij}$ for any pair of nodes. The designer needs to select prices, possibly in a state-dependent way (depending on the configuration of drivers across locations), aiming to maximize some desired objective such as revenue or social welfare. Since such systems tend to operate in fast timescales, we ignore the initial mixing time and focus on the steady-state performance of the resulting processes. In queueing-theoretic terms, the prices create an alternative Markov Decision Process (MDP) whose arrival rates are *thinned* via removing part of the demand; the goal is to create the MDP that optimizes the desired objective.

In this model, we derive a general approximation framework. The approximation ratio of our approach is $1+n/m$: asymptotically optimal as ratio of drivers per location increases and very close to 1 in the real-system parameters (there are typically significantly more drivers than locations). Notably, our pricing policy is state-independent (it outputs only one price for each pair of locations) but the guarantee stands even against state-dependent policies. Our framework applies to a large class of objective functions including throughput, welfare, revenue (under a regularity distributional assumption common in the revenue management literature), Ramsey pricing (max. revenue subject to lower bound on welfare). It also extends to constrained pricing settings such as cases where the prices need to come from some discrete set and to various other rebalancing controls such as deciding which driver to match to a particular customer and allowing for empty-vehicle rebalancing. Finally, our results apply generally to optimization in closed queueing networks (where the number of units remains unaltered), even outside the ridesharing application. We elaborate on these results in Chapter 5.

Technical highlight. Our framework which we term *Elevated Flow Relaxation* is based on solving a convex relaxation of the problem and deriving the approximation ratio via a three-step argument. In revenue management, it is easier to express the objectives in terms of quantiles associated to prices (percentage of demand that has value higher than the price) instead of prices. If there was always a driver available to serve any request, then the objective would be concave and as a result we could apply convex optimization techniques to find the optimal price. However, the difficulty arises due to the network supply externalities: each pricing scheme induces a Markov chain that has some probability of driver unavailability in each node. Unfortunately, the resulting system is non-convex with respect to the quantiles (or the prices) and therefore not easily optimizable.

To tackle this issue, we first drop the dependence on the unavailability probability from the objective function. This makes our objective concave but now the solution of the program does not necessarily correspond to some quantiles derived by some pricing scheme (as it does not deal with unavailability). To address this, we add flow conservation constraints which is a necessary condition for the solution to be actually achievable as quantiles of some pricing policy. We finally need to connect the solution of the relaxation to the m-unit system objectives we are interested in. For that, we show three properties: a) this solution is no less than the optimal state-dependent solution, b) this solution can be achieved by an infinite-unit system, and c) the objective of the m-unit system differs to the one of the infinite-unit system by at most a factor of $1 + n/m$.

1.3.2 Contribution V: Efficiency of dynamic learning outcomes

The task of how the agents should behave is often significantly more complicated than what described in the previous section. In the pricing settings we discussed before, the strategic nature of the agents is very simple: if their value is above the price then they make the purchase, otherwise they abstain. This simplicity is, in fact, one of the reasons why pricing is so universally applied. However, in many situations, the decision of what to do is not that clear. Consider the role of an advertiser in a first-price auction: each advertiser bids on the item, the highest bidder is awarded the item, and pays her bid. Let's assume that the utility of the advertiser is quasilinear, i.e. it is equal to the value minus the price if she obtains the item and 0 otherwise. Now clearly, bidding the actual value is not a good choice as, even if she gets the item, she will get utility of 0. As a result, deciding how to bid is a more complicated task that has to do with understanding how other agents behave and what is the price that is needed for her to win. Fortunately for the advertisers, the online nature of the setting enables them to obtain access to past data and see what bids worked well and what did not. As a result, they can employ online learning techniques, for example the ones discussed in Section 2, to ensure that they have good performance against the best possible fixed bidding strategy they could have used in hindsight.

Positing that players perform at least as well as what adversarial online learning [1] suggests is an easily satisfiable behavioral assumption with nice properties. It is much weaker than the assumption that they play repeatedly the Nash equlibrium of the classical one-shot version [101] which requires them to obtain perfect beliefs about how other players behave in order to best respond to their actions. It is empirically supported as advertising actions can be rationalized via this learning behavioral assumption [102]. Finally, when the same advertisers compete for the same items, the performance of no-regret learning outcomes compares well to a socially effective solution for a large class of games [29, 116, 123].

[1]It is important to use adversarial online learning instead of stochastic as the rewards depend on actions of other agents; assuming that the latter behave in an i.i.d. manner is unrealistic.

Main question. The condition that all players and items remain the same across time points to a significant issue with all the results establishing efficiency under strategic behavior: the underlying settings are never the same across time. In ad-auctions, advertisers may change their value for different keywords based on recent trends or marketing decisions. In packet routing, when a video conference ends, the configuration of data transmission alters. In transportation, when people switch employments or take vacations, similar changes in the routing patterns arise. The efficiency guarantees for learning dynamics improved the relevance of the so called Price of Anarchy guarantees (beyond the restricted notion of Nash equilibria). However, the requirement that the setting is static across time sheds doubt on the applicability of these quarantees. Addressing this issue, we pose the following question:

Are the efficiency guarantees under strategic behavior robust to the frequent changes in dynamically evolving environments?

To tackle this question, in joint work with Vasilis Syrgkanis and Éva Tardos [94], we introduce a dynamic population model parameterized by how rapid the churn of turnover is. More concretely, we have a set of n players and, at each round, every player departs independently with turnover probability p; once this happens, the player is replaced by a new player with arbitrary valuation. This means that, at every round in expectation $p \cdot n$ players leave the system.

The challenge is that a particular player's departure may affect the benchmark solution of multiple different agents. The analysis behind the static efficiency guarantees relies on the fact that agents have no regret for not sticking to the most profitable fixed item (we call this their *favorite item*). Consider a setting with unit-demand advertisers in multi-item auctions (advertisers get no additional utility from getting more than one item). If one advertiser a leaves, then another advertiser b, eyeing for a's previous item, may switch their focus to that. This can create a domino effect with another advertiser c wanting to switch to b's previous item (augmenting path in a bipartite matching). This example creates the impression that, when a player departs, all others need to reinitialize their learning algorithms to target their new favorite items which is problematic for two reasons. First, advertisers need to learn when departures happen while, in ad-auctions, they typically do not even know who the other advertisers are. Second, this reinitialization needs to happen every time that a departure occurs which means that the previous guarantees would only extend if departures happen very sporadically ($p \ll 1/n$) which is not the case in modern platforms.

Result. Countering these intuitions, we show that the efficiency guarantees are robust to high rates of turnover where a constant fraction of the population changes every single round. This result comes through two important techniques. First, many classical online learning algorithms guarantee a stronger notion than regret (shifting regret) that compares to a sequence of benchmark actions instead of the best fixed action; this allows them to seamlessly adapt to changes without needing to reinitialize their algorithms. Second, in many settings, there exist benchmark solutions that are approximately optimal and significantly more stable to agents' departures. This enables us to obtain efficiency guarantees that only lose compared to the static case a minor extra factor due to the stability and gracefully degrade with the turnover probability p allowing for good efficiency guarantees even if p is a constant independent of the number of agents. Applying the above framework to online advertising and routing, we show efficiency guarantees that are robust to the population being dynamically evolving.

Technical highlight. The key technical contribution of this work lies in establishing that the underlying optimization problems of many important game settings enjoy stable sequences of approximately optimal solutions. We provide two different techniques on how such sequences can be identified. First, we show that greedy algorithms with appropriate tie-breaking often come with such stability properties. In that direction, we show that a sticky version of the greedy bipartite matching algorithm provides a stable version of an approximately optimal solution allowing us to obtain efficiency guarantees for ad-auctions with unit-demand bidders. Second, we make a connection between stable solution sequences and joint differentially private solutions. The latter guarantee that the output for any particular user cannot be drastically altered by a change in one coordinate of the input. Connecting this to stability, we provide efficiency guarantees for routing and multi-item auctions.

1.3.3 Contribution VI: A fairness view on online learning

The final facet of this thesis involves the societal context in which platforms operate. Their decisions affect multiple different entities and it is therefore important to understand undesired ethical repercussions they may cause. For instance, targeting ads to particular populations based on irrelevant attributes, such as race, may reinforce stereotypes harmful to society [122, 8]. Similarly a routing platform should try to ensure that the exploration that is necessary for the learning process is not suffered by, say, only minority populations [24, 109].

We focus on group fairness; to obtain a better idea about such notions, we discuss the *equalized odds* notion introduced by Hardt et al. [65]. Consider the task of providing loans; a bank ideally wishes to provide loans to people who will eventually repay them (those with a positive label) rather than to ones who will default (those with a negative label). However, not having access to the true labels, the bank makes mistakes in both directions, either giving loans to people who end up defaulting (false positives) or denying loans to ones who would have returned them if given the opportunity (false negatives). The notion of equalized odds, in its simpler form, imposes that, with multiple populations, the false negative rates (percentage of people with positive label who were denied) is equal among different populations and the same holds for false positive rates (defined analogously). This notion was popularized by a recent debate regarding potential bias of machine learning risk tools for criminal recividism [7, 40, 78, 41].

Although such notions offer a way to reason about the effect of discrimination in decision-making, they largely disregard that data are acquired in an online manner and are not i.i.d. Applications such as online advertising, recommender systems, medical trials, and image classification all require decisions to be made sequentially. The corresponding labels are not identical across time and are affected by the economy, recent events, etc. Similarly labels are not independent across rounds – if a bank offers a loan then this decision can affect whether the loanee or their environment will be able to repay future loans thereby affecting future labels [89]. Moving beyond the batch setting introduces important trade-offs that should be better understood.

Main question. To understand the effect of adaptivity in non-discrimination, we revert to the classical model for non-i.i.d. adaptive decisions, the adversarial online learning setting. The most fundamental version of this setting (experts setting) revolves around the question: *Given a class $\mathcal{F}$ of predictors , how can we* *make online predictions that perform as well as the best predictor in $\mathcal{F}$.* In Section 1.2.1, we revisited this setting where predictors corresponded to alternative actions.

To study the effect of adaptivity in online decision-making, in joint work with Avrim Blum, Suriya Gunasekar, and Nati Srebro [28], we ask the most basic extension of the above question in settings where non-discrimination is an issue:

Given a class $\mathcal{F}$ of ***individually fair predictors****, how can we* ***fairly*** *combine them adaptively to perform as well as the best predictor in $\mathcal{F}$?*

The assumption that predictors are individually non-discriminatory (or fair) is a strong assumption and makes the task trivial when the input is i.i.d., e.g. in the batch setting where the algorithm is given labeled examples and wishes to perform well on unseen examples drawn from the same distribution. This happens because the algorithm can learn the best predictor from the labeled examples and then follow it (since this predictor is individually fair, the algorithm does not exhibit discrimination). This assumption enables us to understand the potential overhead that adaptivity introduces and significantly strengthens any impossibility result. Moreover, we can assume that predictors have been individually vetted to satisfy the non-discrimination desiderata – we therefore wish to understand how to efficiently compose these non-discriminatory predictors while preserving non-discrimination. Finally, this question does not take position on what is the right notion of non-discrimination or fairness and can be applied to any group fairness notion.

Result. We address this question for two different notions of non-discrimination. Our first result is regarding the notion of equalized odds that we discussed before. Surprisingly, we show that there is a fundamental trade-off between performance and equalized odds when adaptivity comes to the picture. We show that no algorithm that achieves the no-regret property, can guarantee equalized odds even within an approximation factor (of, say, 20%) – this holds even for algorithms using the group information. In fact, the examples generating this impossibility result are very simple with just two phases of i.i.d. distributions. This issue seems to suggest that fairness notions establishing equality among groups defined in a label-specific way may be arbitrarily disrupted by the order in which the examples arise. Our second result focuses on achieving this preservation of non-discrimination with respect to the natural requirement of achieving equal average loss among the two groups (regardless if this comes from false negative or false positive examples). Despite proving impossibility results for algorithms that do not use the group information, we show a group-aware algorithm that does achieve the desired guarantee.

Technical highlight. To obtain the positive result for the notion of equal accuracy, we make an interesting learning-theoretic connection shedding light on equality-based fairness notions. Generally, no-regret algorithms guarantee that the average performance of the algorithm is no worse than the average performance of the best predictor. Interestingly, there is a class of algorithms (including the classical multiplicative weights algorithm) where the opposite is also true: the average performance of the algorithm is also no better than the average performance of the best predictor [59]. As a result, the average performance of the algorithm at each of the groups is approximately equal to the average performance of the best predictor at the group. Since we assume that the predictors are individually non-discriminatory with respect to the average loss, it means that the average performance of the predictor at the one group is the same across all groups which then establishes the positive result. This property is, to a large extent, essential – in fact, we show that algorithms with stronger guarantees (shifting regret algorithms) suffer from impossibility results.

1.4 Roadmap of this thesis

This introduction chapter served as an initial exposition to the context of this thesis and the main contributions in it. In the following chapters, we elaborate on each of the contributions in the thesis. The chapters are intended to be self-contained in order to be able to be read on their own, though we point to connections to previous chapters where appropriate.

For each chapter x, our aim is to convey the following information in a similar structure. We first start with an introductory part introducing the reader to the setting; we advise the reader to also refer to the corresponding subsection in Chapter 1 as not all points are repeated in this part. Subsequently, in Section $x.1$, we provide background technical information that is necessary for the readability of the chapter. By Section $x.2$, we provide the details of the particular model and desiderata that we wish to achieve in the chapter. Section $x.3$ serves as an exposition of what can go wrong and why classical approaches fail to address the question under investigation. Sections $x.4$ and $x.5$ describe the main result of the chapter as well as an additional result (either a warm-up or a follow-up to the main result). Finally, Section $x.6$ aims to put the particular work in the broader context, discusses other works in the area, elaborates on particular assumptions, and points to important open questions.

CHAPTER 2
MITIGATING EXPLORATION IN ONLINE LEARNING

Maybe the biggest bottleneck in online decision-making is related to the insufficient information regarding the system at hand. In complex systems, the decision-maker often deals repeatedly with a similar task, trying to decide across a set of different alternatives. The challenge is to make effective decisions despite not knowing initially any information about the system and only receiving partial feedback determined by the selected action. We focus on settings where the reward or loss of different alternatives does not follow nice stochastic patterns (e.g. it is not i.i.d. across time). This non-stochasticity is often due to interactions of multiple agents whose decisions affect the performance of different alternatives.

Originated by game-theoretic considerations in multi-agent dynamics [25, 64], adversarial online learning emerged as a way to deal with online decision-making without imposing any distributional assumption on the input. This powerful framework provides a robust way to balance exploring different alternatives and exploiting ones that have been profitable in the past. Surprisingly, even without any prior information about the system and even when losses are adversarially selected, these techniques can guarantee performance asymptotically as good as the one of the best alternative in hindsight. Intuitively, despite the fact that the learner is initially clueless about different options, she can soon realize that some actions perform well, and can therefore start selecting them. In Chapter 6, we will see that this property has important consequences regarding the efficiency of complex systems with selfish participants.

In this chapter, we address an important limitation in current adversarial online learning techniques when applied in realistic settings where the learner only has access to partial feedback. In particular, although classical partial-feedback online learning techniques achieve asymptotically good performance, the rate in which this is achieved is relatively large, which is ineffective in practice. This issue arises because these approaches need to revert to over-exploration to deal with the non-stochasticity of the environment, even when there exist actions that are really good (and therefore learning to follow them should occur easier). We will show how to mitigate this phenomenon when there exists one such good action (with small loss) without significantly sacrificing the worst-case performance of the system. This is one example of obtaining data-dependent guarantees for online decision-making that can utilize some well-behaved structure in the data while being robust to this structure not holding. In the subsequent two chapters, we will see two more examples where such data-dependent guarantees can arise.

2.1 Preliminaries on adversarial online learning

Online learning setting. We first introduce the basic online learning setting, which describes the framework in which the sequential decisions are made. The decision-maker or learner has access to a set of d alternatives that we will refer to as *arms* or *actions* $a = 1, \dots, d$. At round $t = 1, \dots, T$, the following process occurs:

1. The learner selects a probability distribution $p^t \in \Delta(d)$ over the d possible arms; this is such that $\sum_{i=1}^{d} p_a^t = 1$.
2. The adversary then selects losses $\ell^t = (\ell_1^t, \dots, \ell_d^t)$ where $\ell_a^t \in [0, 1]$ denotes the loss of action a at round t and is assumed to lie in $[0, 1]$.

3. The learner then draws action $A(t) \sim p^t$ from the distribution p^t she committed to and suffers the loss of the selected action $\ell^t_{A(t)}$.
4. The learner observes feedback about the losses based on a feedback model.

In the full feedback model (experts setting), the learner observes the loss of all the actions $\{\ell^t_a\}_{\forall a}$ regardless what she selected. In the bandit feedback model, she only observes feedback only for the selected action $\ell^t_{A(t)}$. We will focus on a general feedback model interpolating between these two extremes.

Graph-based feedback model. In this chapter, we focus on a feedback model suggested by Mannor and Shamir [96] where the learner receives partial feedback based on an undirected feedback graph $G(t)$ that possibly varies across rounds. The learner observes the loss $\ell^t_{A(t)}$ of the selected arm $A(t)$ and, in addition, she also observes the losses of all arms connected to the selected arm $A(t)$ in $G(t)$. More formally, she observes the loss $\ell^t_{a'}$ for all the arms $a' \in N^t_{A(t)}$ where N^t_a denotes the set containing arm a and all neighbors of a in $G(t)$ at round t. The *full feedback setting* and the *bandit feedback setting* are special cases of this model where the graphs $G(t)$ are the complete and the empty graph respectively for all rounds t.

We allow the feedback graph $G(t)$ to change each round t, but assume that the graph $G(t)$ is known to the player before selecting her distribution p^t. This model also includes the *contextual bandits* problem of [12, 87] as a special case, where each round the learner is presented with an additional input x^t, the context. In this contextual setting, the learner is offered d policies, each suggesting an action depending on the context, and each round the learner can decide which policy's recommendation to follow. To model this with our evolving feedback graph model, we use the policies as nodes, and connect two policies with an edge in

$G(t)$ if they recommend the same action in the context x^t of round t.

Regret. The goal of the learner is to minimize the loss of the algorithm. On its own, the loss of the algorithm is not providing enough insight of whether the algorithm is good or not. The loss of the algorithm may be large because the algorithm selects suboptimal arms, but it may also be large because no arm has good performance. As a result, to evaluate how well the algorithm is doing, we typically focus on the so called regret against an appropriate benchmark. The traditional notion of regret compares the performance of the algorithm to the best fixed action f in hindsight. For an arm f we define regret as:

$$Regret(f) = \sum_{t=1}^{T} \left[\ell^t_{A(t)} - \ell^t_f \right].$$

To evaluate performance, we consider regret against the best arm:

$$Regret = \max_f Regret(f)$$

Note that both $Regret(f)$ and $Regret$ are random variables, depending on the randomness in the algorithm.

A slightly weaker notion of regret is pseudoregret (c.f. [35]), which compares the expected loss of the algorithm to the expected loss of any fixed arm f, fixed in advance and not in hindsight. More formally, this notion of expected regret is:

$$PseudoReg = \max_f \mathop{\mathbb{E}}_{A(1)\ldots A(t)} [Regret(f)]$$

This is weaker than the expected regret $\mathbb{E}_{A(1)\ldots A(t)}[Regret] = \mathbb{E}_{A(1)\ldots A(t)}\left[\max_f Regret(f)\right]$.[1].

[1]To see the difference, consider n arms that are similar but have high variance. Pseudoregret compares the algorithm's performance against the expected performance of arms, while regret compares against the "best" arm depending on the outcomes of the randomness. This difference can be quite substantial, like when throwing n balls into n bins the expected load of any bin is 1, while the expected maximum load is $\Theta(\log n / \log\log n)$.

We aim for an even stronger notion of regret, guaranteeing low regret with high probability, i.e. for all $\delta > 0$ with probability $1 - \delta$, instead of only in expectation, at the expense of a logarithmic dependence on $1/\delta$ in the regret bound for any fixed δ. Note that any high-probability guarantee concerning $Regret(f)$ for any fixed arm f with failure probability δ' can automatically provide an overall regret guarantee with failure probability $\delta = d\delta'$. A high-probability guarantee on low *Regret* also implies low regret in expectation.[2]

2.2 Small-loss guarantees with partial feedback

Classical regret guarantees with an appropriate feedback dimension. Adversarial online learning aims to achieve the so called vanishing regret. This means that regret scales sublinearly with the time horizon T. Hence the average regret is vanishing as time grows large; this is typically referred to as the *no-regret property*. In the full feedback case, there are very simple and natural algorithms achieving this property with the regret scaling as a function of $\sqrt{T \log(d)}$. Examples include multiplicative weights [54] and follow the perturbed leader [72].

More recently, this property was satisfied even at the absence of full feedback, scaling with an appropriate feedback dimension of the feedback model. In bandit feedback [12] the regret is of the order of $\sqrt{dT}$ scaling with the number of arms d. This dependence on d stems from the fact that, even with i.i.d. losses for each arm, the learner may need to select all arms enough times to identify the best performing. In the general graph-based feedback setting, the dependence on d can

[2]If the algorithm guarantees regret at most $B \log(1/\delta)$ with probability at least $(1 - \delta)$ for any $\delta > 0$, then we can obtain the expected regret bound of $O(B)$ by upper bounding the expected regret by the integral $\int_0^\infty x \cdot \mathbb{P}(Regret > x)dx$.

be replaced by the cardinality of the largest independent set $\alpha(G)$ of the feedback graph G [6], also referred to as *independence number*. Intuitively, this dependence is necessary as it is plausible that all arms not lying on the largest independent set have always high loss. As we cannot select them often, the setting reduces to bandit feedback on the α nodes lying on the largest independent set.

Small-loss regret bounds. One issue with the above guarantees is that they scale with the time-horizon T. When the input is not i.i.d. and we do not receive full feedback, the only way to have an up-to-date view of how different arms perform is by exploring all of them (including suboptimal ones). However, doing so, we select suboptimal actions with big enough probability every single round which, inescapably, leads to regret guarantees that scale with T (this is formally described in the next section). This is particularly undesirable in settings where there exist some almost perfect actions with really small loss (that is significantly less than T) – in that case, the input has a well-behaved structure as this action is more easily identifiable. However, the effort to keep an up-to-date view of the world leads to exploring suboptimal actions often despite the existence of an almost perfect action. This over-exploration is a significant roadblock towards employing online learning algorithms in practice.

To address this over-exploration, one approach is to aim for regret guarantees that prohibit it, i.e. any algorithm that over-explores cannot satisfy them. One such guarantee is the *small-loss* regret bounds, where the regret of the algorithm needs to scale with the loss of the best arm instead of the time horizon. To achieve these guarantees, we first focus on the notion of approximate regret (c.f. [53]), which is a multiplicative relaxation of the regret notion. We define *ϵ-approximate* *regret* for a parameter $\epsilon > 0$ and an arm f as

$$ApxReg(f,\epsilon) = (1-\epsilon)\sum_{t=1}^{T} \ell^t_{A(t)} - \sum_{t=1}^{T} \ell^t_f.$$

We prove bounds on $ApxReg(f,\epsilon)$ in high probability and in expectation, and use these to provide small-loss regret bounds by tuning ϵ appropriately via an approach that is often used in the literature in achieving classical regret guarantees and is referred to as *self-confidence* [13]. In Chapter 6, we will also see that approximate regret is also often useful in its own sake.

Typically, approximate regret bounds depend inversely on the parameter ϵ; to derive small-loss regret bounds, this needs to be appropriately tuned over time. For instance, in classical full feedback algorithms such as multiplicative weights [54] or follow the perturbed leader [72], the expected approximate regret is bounded by $O(\log(d)/\epsilon)$ and therefore setting $\epsilon = \sqrt{\log(d)/T}$, one obtains the classical $O(\sqrt{T\log(d)})$ uniform bounds. If we knew $L^\star$, the loss of the best arm at the end of round T, one could set $\epsilon = \sqrt{\log(d)/L^\star}$ and get the desired $O\left(\sqrt{L^\star\log(d)}\right)$ guarantee. Of course, $L^\star$ is not known in advance, and depending on the model of feedback, may not even be observed either. To overcome these difficulties, we can make the choice of ϵ depend on $\widehat{L}$, the loss of the algorithm instead, and apply doubling trick: start with a relatively large ϵ, hoping for a small $\widehat{L}$ and halve ϵ when we observe higher losses. This combines doubling trick with the idea of the so called self-confident online learning approach [13].

2.3 Classical reduction cannot give small-loss guarantees

Importance sampling and roadblock with variance. The classical way to translate such full-feedback small-loss results to partial feedback fails as they rely on observing often the losses of all arms. The full-feedback case is very well understood and we have many algorithms that achieve approximate regret of $O(\log(d)/\epsilon)$. The natural way to extend these results to partial feedback is to try to create estimated losses that can be created via the available feedback and *behave as if they were the actual losses*. This can be achieved, for example, through what is called importance sampling or inverse propensity weighting. In classical importance sampling, the estimated loss of an arm is equal to its actual loss divided by the probability of it being observed, if the arm is observed, and 0 otherwise. This makes the estimator unbiased as the expected estimated loss of any arm is equal to its actual loss. Such estimators lie in the heart of the reductions providing regret guarantees for bandit feedback [12] as well as graph-based feedback [6]. In the graph-based feedback model, we acquire information for all arms observed and not only for the ones played. As a result, importance sampling is applied via dividing the loss of the arm when observed by its probability of observation:

$$\tilde{\ell}_a^t = \frac{\ell_a^t}{\sum_{a' \in N_a^t} p_{a'}^t} \mathbf{1}_{A(t) \in N_a^t}.$$

Let's see formally how such a reduction looks like to understand the first roadblock. We wish to show that the approximate regret scales sublinearly to the loss of the best action $\sum_t \ell_f^t$ and only depends on the maximum independence number α across all graphs $G(t)$, having only a logarithmic dependence on the number of arms d. For bandit feebdack, $\alpha = d$; the roadblock applies even then.

Suppose that we apply the above reduction with a full information algorithm $\mathcal{A}$ with approximate regret $\log(d)/\epsilon$ for parameter ϵ. Since the estimated losses do not lie in $[0, 1]$ and since the algorithm is applied on the estimated losses, the resulting approximate regret scales with the maximum estimated loss $\max_{a,t} \tilde{\ell}_a^t$.

Starting from the loss of the algorithm, we can obtain:

$$\begin{aligned}
(1-\epsilon)\,\mathbb{E}\left[\sum_t \ell^t_{A(t)}\right] &= (1-\epsilon)\,\mathbb{E}\left[\sum_t \sum_i p^t_a \tilde{\ell}^t_a\right] && \text{as } \mathbb{E}\left[\tilde{\ell}^t_a\right] = \ell^t_a \text{ on all arms played.}\\
&\le \mathbb{E}\left[\sum_t \tilde{\ell}^t_f + \left(\max_{a,t} \tilde{\ell}^t_a\right)\cdot \frac{\log(d)}{\epsilon}\right] && \text{by the low approx regret of } \mathcal{A}.\\
&\le \sum_t \mathbb{E}\left[\ell^t_f + \left(\max_{a,t} \tilde{\ell}^t_a\right)\cdot \frac{\log(d)}{\epsilon}\right] && \text{as } \mathbb{E}\left[\tilde{\ell}^t_f\right] = \ell^t_f
\end{aligned}$$

As a result, if we could get an upper bound on the quantity $\max_{a,t} \tilde{\ell}^t_a$ then we would be able to derive an approximate regret guarantee despite the partial feedback. Unfortunately, this quantity can be arbitrarily large as the probability of observing any arm can be aribtrarily low (and it appeears in the denominator of the estimator). This poses a major roadblock towards in the black-box reduction from a classical full feedback algorithm.

Mixing uniform action-distribution and new roadblock. To deal with this, typical partial information algorithms, such as EXP3 [12] or EXP3-DOM [6], mix the resulting distribution with a uniform action distribution, guaranteeing a lower bound on the probability of being observed and therefore an upper bound on the range of estimated losses. More formally, if the full-feedback algorithm suggests a distribution $\tilde{p}^t$, the resulting probability for arm a is $p^t_a = (1-\theta)\cdot \tilde{p}^t_a + \theta \cdot (1/d)$ for a new parameter θ. As a result, this guarantees an upper bound on $\max_{a,t} \tilde{\ell}^t_a \le d/\theta$.

However, when doing so, the algorithm's performance suffer from at least an extra $(\theta/d)\cdot(\max_a \sum_t \ell^t_a)$ as with this much probability the algorithm selects the arm with the highest loss (over-exploration). As a result, the performance of the algorithm scales with the worst arm which may have loss of 1 every single round and be linear on $\theta \cdot T$ instead of depending on the cumulative loss $L^\star$ of the best arm which may besignificantly smaller when the input is well behaved. It is easy to see that when $\max_a \sum_t \ell_a^t = T$, there is no way to set the mixing parameter θ that will avoid introducing a dependence on the time horizon T. Intuitively, since the added mixing makes the algorithm select badly performing arms, this approach results in uniform regret bounds and not small-loss guarantees.

2.4 Main result: General reduction to partial feedback

Instead of mixing with a uniform action distribution, we use an alternate technique, first proposed by Allenberg et al. [5] in the context of the Multiplicative Weights algorithm for bandit feedback. We set a threshold γ and in each round neither play nor update the loss of arms with probability below this threshold. We refer to such arms as (temporarily) *frozen*. We note that frozen arms may get unfrozen in later rounds, if other arms incur losses, as we update frozen arms assuming their loss is 0. The resulting estimator for the loss of an arm is no longer unbiased since the estimated loss of frozen arms is 0. However, crucially the estimator is unbiased for the selected arms and negatively biased for all arms; this allows us to extend the regret bound of the full-feedback algorithm. When freezing arms, we need to normalize the probabilities of other arms so that they form a probability distribution. To obtain ϵ-approximate regret guarantees, the total probability of all frozen arms should be at most $\epsilon' = \Theta(\epsilon)$. Allenberg et al. [5] guarantee this for the bandit feedback setting by selecting $\gamma = \epsilon'/d$ resulting in a dependence on the number of arms in the approximate regret bound.

In this section we extend this technique in three different ways:

- We obtain small-loss learning algorithms for the case of feedback graphs, where the regret bound depends on the maximum independence number $\alpha = \max_t \alpha(G^t)$, instead of d (number of nodes).

- We achieve the above via a black-box reduction using any full information algorithm, not only via using the Multiplicative Weights algorithm.
- We provide a small-loss guarantee that holds with high probability and not only in expectation.

Seeking for bounds that are only a function of the size $\max_t \alpha(G^t)$, and have no dependence on the number of arms, we introduce a novel double-threshold freezing technique. At each round t, we first freeze arms that are observed with probability less than some threshold γ. We show (Claim 2.1) that the total probability frozen at this initial step is at most $\alpha(G^t)\gamma$. However, freezing an arm may cause a snowball effect, decreasing the probability that its neighbors are observed. This can propagate and cause additional arms to be observed with probability less than γ, violating the upper bound on the estimated loss. To bound the total probability frozen in the propagation steps as a function of $\alpha(G^t)$ while maintaining a lower bound on the probability of observation for the played arms, we recursively freeze all arms with observation probability smaller than $\gamma' = \gamma/3$. We show in Claim 2.2 that the total probability frozen in the recursive process is at most 3 times the total probability frozen in the initial step.

We proceed by providing the algorithm (Algorithm 1), the crucial lemma that enables improved bounds beyond bandit feedback (Lemma 2.1), and the black-box guarantee. For clarity of presentation we first provide the approximate regret guarantee in expectation (Theorem 2.1) and then show its high-probability version (Theorem 2.2), in both cases assuming that the algorithm has access to an upper bound of the maximum independence number α as an input parameter.

In Theorem 2.3 we provide the small-loss version of the above bound without explicit knowledge of this quantity.

Algorithm 1: Double-Threshold Freezing Algorithm

```
Require: Full-feedback algorithm $\mathcal{A}$, an upper bound on the size of maximum independent sets $\alpha$, number of arms $d$, learning parameter $\epsilon'$.
1: Initialize $\tilde{p}_a^1$ for arm $a$ based on the initialization of $\mathcal{A}$ and set $t = 1$.
2: for $t = 1$ to $T$ do
3:     Initial step: Freeze arms whose observation probability is below $\gamma = \epsilon'/(4\alpha)$ to obtain: $F_0^t = \left\{a : \sum_{a' \in N_a^t} \tilde{p}_{a'}^t < \gamma\right\}$.
4:     Propagation steps: Recursively freeze arms if their probability of being observed by unfrozen arms is below $\gamma' = \gamma/3$ to obtain $F^t = \bigcup_{k \geq 0} F_k^t$ where:
       $$F_k^t = \left\{ a \notin \left( \bigcup_{m=0}^{k-1} F_m^t \right) : \sum_{a' \in \left(N_a^t \setminus \bigcup_{m=0}^{k-1} F_m^t\right)} \tilde{p}_{a'}^t < \gamma' \right\}$$
5:     Normalize probabilities of unfrozen arms so that they form a distribution:
       $p_a^t = 0$ if $a \in F^t$ and $p_a^t = \frac{\tilde{p}_a^t}{1 - \sum_{a' \in F^t} \tilde{p}_{a'}^t}$ otherwise.
6:     Draw arm $A(t) \sim p^t$ and incur loss $\ell_{A(t)}^t$.
7:     Compute estimated losses: $\tilde{\ell}_a^t = \frac{\ell_a^t}{\sum_{a' \in N_a^t} p_{a'}^t}$ if $a \in N_{A(t)}^t \setminus F^t$ and $\tilde{\ell}_a^t = 0$ otherwise.
8:     Update $\tilde{p}_a^{t+1}$ using full information algorithm $\mathcal{A}$ with loss $\tilde{\ell}^t$ for round $t$.
9: end for
```

Lemma 2.1. At every round t, the total probability of frozen arms is at most ϵ': $\sum_{a \in F^t}^t \tilde{p}_a^t \leq \epsilon'$, and hence any non-frozen arm a increases its probability due to freezing by a factor of at most $(1 - \epsilon')$.

The proof of the lemma follows from understanding how much probability is frozen in the initial steps and how much is frozen during the propagation steps. These are bounded in the two following claims.

Claim 2.1. The total probability frozen in the initial step is $\sum_{a \in F_0^t} \tilde{p}_a^t \leq \alpha(G^t)\gamma$.

Proof. Let S^t be a maximal independent set on F_0^t. Since the independent set is maximal, every node in F_0^t either is in S^t or has a neighbor in S^t, so we obtain:

$$\sum_{a \in F_0^t} \tilde{p}_a^t \leq \sum_{a \in S^t} \sum_{a' \in \left(N_a^t \cap F_0^t\right)} \tilde{p}_{a'}^t < \alpha(G^t) \cdot \gamma.$$

where the last inequality follows since there are at most $\alpha(G^t)$ nodes in S^t and, as they are frozen, the probability of being observed is at most γ for each. ■

Claim 2.2. The total probability frozen in the propagation steps is bounded by three times the total probability frozen at the initial step. More formally:

$$\sum_{a \in \bigcup_{k \geq 1} F_k^t} \tilde{p}_a^t \leq 3 \sum_{i \in F_0^t} \tilde{p}_a^t.$$

Proof. The purpose of the lower threshold γ' in line 4 is to limit the propagation of frozen probability. Consider an arm a frozen on step $k \geq 1$. Since arm a was not frozen at step 0, the initial probability of being observed by any node of G^t is at least $\gamma = 3\gamma'$. When this arm becomes frozen, it is observed with probability at most γ'. Hence $2\gamma'$ of the original probability stems from arms frozen earlier. Using this, we can bound the probability mass in F_1^t by at most 1.5 times the mass of F_0^t. Further, from these arms at most γ' of the originally at least $3\gamma'$ probability is newly frozen, and hence can affect non yet frozen arms, creating further cascade. We show that the total frozen probability is at most 3 times the probability of nodes in F_0^t. The proof of this fact follows in a way analogous of how the number of internal nodes of a binary tree is bounded by the number of leaves, as any node can have at most 1 parent, while having 2 children.

More formally, we consider an auxiliary function that serves as an upper bound of the left hand side and a lower bound of the right hand side, proving the claim. The claim is focused on a single round t. For simplicity of notation, we drop the dependence on t from the notations, i.e., use $F = \cup_k F_k$ for the set of nodes frozen, $\tilde{p}_a$ for the probability of node a, use G for the graph, and E for its edge-set. Let $F_{\geq 1} = \bigcup_{k \geq 1} F_k^t$. We order all nodes in F based on when they are frozen. More formally, if $a \in F_m$ and $a' \in F_k$ with $m < k$ then $a \prec a'$. This is a partial ordering as $\prec$ does not order nodes frozen at the same iteration of the recursive freezing. We now introduce the heart of the auxiliary function which lies in the sum of the products of probabilities $\tilde{p}_a \cdot \tilde{p}_{a'}$ along edges (a, a') with $a \prec a'$, such that $(a, a') \in E$, i.e.

$$\sum_{\substack{a \in F, a' \in F_{\geq 1}, a \prec a' \\ (a,a') \in E}} \tilde{p}_a \tilde{p}_{a'} \tag{2.1}$$

To lower bound the quantity in (2.1), we sum over a' first. Node a' was not in F_0 so its neighborhood has a total probability mass of at least $\gamma = 3\gamma'$. By the time a' is frozen, the remaining probability mass is less than γ', so a total probability mass of at least $2\gamma'$ must come from earlier frozen neighbors.

$$\sum_{\substack{a \in F, a' \in F_{\geq 1}, a \prec a' \\ (a,a') \in E}} \tilde{p}_a \tilde{p}_{a'} = \sum_{a' \in F_{\geq 1}} \tilde{p}_{a'} \cdot \left[\sum_{\substack{a \in F, a \prec a' \\ (a,a') \in E}} \tilde{p}_a \right] \geq \sum_{a' \in F_{\geq 1}} \tilde{p}_{a'} \cdot 2\gamma'$$

To upper bound the quantity in (2.1), we sum over a first, and separate the sum for $a \in F_0$ and $a \in F_{\geq 1}$. Nodes $a \in F_0$ have total probability of less than $\gamma = 3\gamma'$ in their neighborhood, as they are frozen in line 3 of the algorithm. Nodes $a \in F_{\geq 1}$ have at most γ' probability mass left in their neighborhood when they become frozen, thus at most this much total probability contributes to the products with neighbors later in the ordering.

$$\sum_{\substack{a\in F, a'\in F_{\geq 1}, a<a'\\(a,a')\in E}} \tilde{p}_a \tilde{p}_{a'} = \sum_{i\in F_0} \tilde{p}_a \left[\sum_{\substack{a'\in F_{\geq 1}, a<a'\\(a,a')\in E}} \tilde{p}_a \right] + \sum_{a\in F_{\geq 1}} \tilde{p}_a \left[\sum_{\substack{j\in F_{\geq 1}, a<a'\\(i,j)\in E}} \tilde{p}_{a'} \right] \leq \sum_{i\in F_0} \tilde{p}_a \cdot 3\gamma' + \sum_{a\in F_{\geq 1}} \tilde{p}_a \cdot \gamma'$$

The above lower and upper bounds imply that:

$$2\gamma' \sum_{a\in F_{\geq 1}} \tilde{p}_a \leq 3\gamma' \sum_{a\in F_0} \tilde{p}_a + \gamma' \sum_{a\in F_{\geq 1}} \tilde{p}_a.$$

Hence we obtain the claimed bound (reintroducing the round t in the notation):

$$\sum_{a\in \bigcup_{k\geq 1} F_k^t} \tilde{p}_a^t \leq 3 \sum_{a\in F_0^t} \tilde{p}_a^t$$

■

Proof of Lemma 2.1. We first consider the arms frozen due to the γ-threshold (line 3 of the algorithm). Claim 2.1 shows that the total probability frozen in the initial step is bounded by $\sum_{a\in F_0^t} \tilde{p}_a^t \leq \alpha(G^t)\gamma$. We then focus on the arms frozen due to the recursive γ'-threshold (line 4 of the algorithm). Claim 2.2 bounds the total probability frozen in the propagation processs by three times the total probability frozen in the initial step. Combining the two Claims, we obtain:

$$\sum_{i\in F^t} \tilde{p}_i^t = \sum_{i\in F_0^t} \tilde{p}_i^t + \sum_{i\in \bigcup_{k\geq 1} F_k^t} \tilde{p}_i^t \leq \alpha(G^t)\gamma + 3\alpha(G^t)\gamma = 4\alpha(G^t)\gamma \leq \epsilon'.$$

The lemma then follows from the normalization step of the algorithm (line 5). ■

Bounding pseudoregret. We are now ready to prove our first result: a bound for learning with partial feedback based on feedback graphs. We first provide the guarantee for approximate pseudoregret in expectation. We assume both the learning rate ϵ as well as an upper bound α on the size of the independent sets are given as an input. At the end of this section, we turn these results into regret guarantees via doubling trick without knowledge of the independence number.

Theorem 2.1. Let $\mathcal{A}$ be any full-feedback algorithm with an expected approximate regret guarantee given by: $\mathbb{E}[ApxReg(f, \epsilon/2)] \le 2L \cdot A(d, T)/\epsilon$ against any arm f, when run on losses in $[0, L]$. The *Double-Threshold Freezing Algorithm* run with learning parameter $\epsilon' = \epsilon/2$ on input $\mathcal{A}$, α, d, has expected ϵ-approximate regret guarantee: $\mathbb{E}[ApxReg(f, \epsilon)] = 48\alpha \cdot A(d, T)/\epsilon^2$.

Proof. The proof follows the classical reduction described in Section 2.3 but uses freezing to deal with the resulting shortcomings in three ways. First, freezing guarantees that the maximum estimated loss is $L = 1/\gamma'$ (since the probability of being observed is at least γ' for any non-frozen arm; else this arm becomes frozen at step 4 of the algorithm). Second, although the estimator is no longer unbiased for all arms, it is unbiased for all non-frozen arms $a \notin F^t$ at all rounds t, i.e. $\mathbb{E}[\tilde{\ell}_a^t] = \ell_a^t$. It is always negatively (optimistically) biased regardless of whether the arm is frozen or not, i.e. $\mathbb{E}[\tilde{\ell}_a^t] \le \ell_a^t$. Finally, the frozen probability is distributed proportionally to the probabilities of non-frozen arms, hence increases the algorithm's loss proportionally. This is in contrast with mixing a uniform action distribution in which case the extra probability is distributed across all arms uniformly, resulting to guarantees that scale with the performance of the worst arm. More formally:

$$
\begin{aligned}
(1-\epsilon)\,\mathbb{E}\left[\sum_t \ell_{A(t)}^t\right] &= (1-\epsilon)\,\mathbb{E}\left[\sum_t \sum_a p_a^t \tilde{\ell}_a^t\right] && \text{as } \mathbb{E}\left[\tilde{\ell}_a^t\right] = \ell_a^t \text{ on all arms played.}\\
&\le \frac{1-\epsilon}{1-\epsilon/2}\,\mathbb{E}\left[\sum_t \sum_a \tilde{p}_a^t \tilde{\ell}_a^t\right] && \text{by Lemma 2.1.}\\
&\le \mathbb{E}\left[\sum_t \tilde{\ell}_f^t + L \cdot \frac{A(d,T)}{\epsilon'}\right] && \text{by the low approx regret of } \mathcal{A}.\\
&\le \sum_t \mathbb{E}\left[\ell_f^t\right] + \frac{A(d,T)}{\gamma' \cdot \epsilon'} && \text{as estimator is negatively biased}\\
&= \sum_t \mathbb{E}\left[\ell_f^t\right] + 48\alpha \cdot \frac{A(d,T)}{\epsilon^2} && \text{using definitions of } L, \gamma', \gamma \text{ and } \epsilon'.
\end{aligned}
$$

The second inequality also uses the fact that $(1-\epsilon) \le (1-\epsilon/2)^2$. ■

Notice that, for the result, it was important to be able to use a freezing threshold $\gamma \propto \epsilon/\alpha$ instead of $\gamma \propto \epsilon/d$ for the above analysis, allowing an approximate regret bound with no dependence on d.

High probability bound. To obtain a high-probability guarantee (and hence a bound on the actual regret, not pseudoregret), we encounter an additional complication since we need to upper bound the cumulative estimated loss of the comparator by its cumulative actual loss. For this purpose, the mere fact that the estimator is negatively biased does not suffice. The estimator may, in principle, be unbiased (if the arm is never frozen), and the variance it suffers can be high, which could ruin the small-loss guarantee. To deal with this, we apply a concentration inequality, comparing the expected loss to a multiplicative approximation of the actual loss. This is inspired by the approximate regret notion, is a quantity with negative mean, and has variance that depends on $1/\epsilon$ as well as the magnitude of the estimated losses which is $1/\gamma'$.

Theorem 2.2. Let $\mathcal{A}$ be any full-feedback algorithm with an expected approximate regret guarantee of: $\mathbb{E}[ApxReg(f, \epsilon/5)] \le 5L \cdot A(d,T)/\epsilon$, against any arm f, when run on losses in $[0, L]$. For any $\delta > 0$ with probability $1-\delta$, the *Dual-Threshold Freezing Algorithm* run with learning parameter $\epsilon' = \epsilon/5$ on input $\mathcal{A}$, α, d, has ϵ-approximate regret: $ApxReg(f, \epsilon) = O\left(\frac{\alpha \cdot \left(A(d,T) + \log(d/\delta)\right)}{\epsilon^2}\right)$.

To prove the theorem, we need the following concentration inequality, showing that the sum of a sequence of (possibly dependent) random variables cannot be much higher than the sum of their expectations conditioned to the past[3]:

[3]The conditional expectations are still random variables depending on past realizations.

Lemma 2.2. Let $(x_t)_{t\geq 1}$ be a sequence of non-negative random variables, s.t. $x_t \in [0,1]$. Let $\mathbb{E}_{t-1}[x_t] = \mathbb{E}[x_t | x_1, \ldots, x_{t-1}]$. Then, for any $\epsilon, \delta > 0$, with probability at least $1 - \delta$

$$\sum_t x_t - (1+\epsilon)\sum_t \mathbb{E}_{t-1}[x_t] \leq \frac{(1+\epsilon)\ln(1/\delta)}{\epsilon}$$

and also with probability at least $1 - \delta$

$$(1-\epsilon)\sum_t \mathbb{E}_{t-1}[x_t] - \sum_t x_t \leq \frac{(1+\epsilon)\ln(1/\delta)}{\epsilon}$$

The proof follows the outline of classical Chernoff bounds for independent variables combined with the law of total expectation to handle the dependence. For completeness, the proof details are provided in Appendix A.1.

Proof of Theorem 2.2. To obtain a high-probability statement, we use Lemma 2.2 multiple times as follows:

1. Show that the sum of the algorithm's losses stays close to the sum of the expected losses.
2. Show that the sum of the expected losses stays close the sum of the expected estimated losses used by the full information algorithm $\mathcal{A}$
3. Show that the sum of the estimated losses of each arm f stays close to the sum of the actual losses.

Starting with the item 1, we use $x_t = \ell^t_{A(t)}$, and note that its expectation conditioned on the previous losses is $m_t = \sum_i p^t_i \ell^t_i$ so we obtain that, for any $\delta', \epsilon > 0$, with probability at least $(1 - \delta')$

$$\sum_t \ell^t_{A(t)} - (1+\epsilon')\sum_t \sum_i p^t_i \ell^t_i \leq \frac{(1+\epsilon')\ln(1/\delta')}{\epsilon'}$$

Next item 3, for a comparator f we use the lemma with $x_t = \tilde{\ell}_f^t$ and its expectation $m_t = \ell_f^t$. Now x_t is bounded by $1/\gamma$ and not 1, so by scaling we obtain that with probability $(1 - \delta')$

$$\sum_t \widetilde{\ell}_f^t - (1 + \epsilon') \sum_t \ell_f^t \leq \frac{(1 + \epsilon') \ln(1/\delta')}{\gamma \epsilon'}$$

Finally, we use the lower bound in the lemma to show item 2: for $x_t = \sum_i p_i^t \widetilde{\ell}_i^t$, the expected losses observed by the full information algorithm, and its expectation $m_t = \sum_i p_i^t \ell_i^t$. Again, since $x_t \in [0, 1/\gamma]$, with probability $(1 - \delta')$,

$$\sum_t \sum_i p_i^t \ell_i^t - (1 + \epsilon') \sum_t \sum_i p_i^t \tilde{\ell}_i^t \leq \frac{(1 + \epsilon') \ln(1/\delta')}{\gamma \epsilon'}$$

Using union bound and $\delta' = \delta/(d + 2)$, all these inequalities hold simultaneously for all $\delta > 0$. To simplify notation, we use $B = \frac{(1+\epsilon') \ln((d+2)/\delta)}{\gamma \epsilon'}$ for the error bounds above.

Combining all the bounds we obtain that

$$\begin{aligned}
\sum_t \ell_{A(t)}^t &\leq (1 + \epsilon') \sum_t \sum_i p_i^t \ell_i^t + B && \text{by item 1 above} \\
&\leq \frac{1 + \epsilon'}{1 - \epsilon'} \left(\sum_t \sum_i \tilde{p}_i^t \ell_i^t + B \right) && \text{by Lemma 2.1} \\
&\leq \frac{(1 + \epsilon')^2}{1 - \epsilon'} \left(\sum_t \sum_i \tilde{p}_i^t \tilde{\ell}_i^t + 2B \right) && \text{by item 2 above} \\
&\leq \frac{(1 + \epsilon')^2}{(1 - \epsilon')^2} \left(\sum_t \tilde{\ell}_f^t + 2B + \frac{A(d, T)}{\gamma \cdot \epsilon'} \right) && \text{by the low approx. regret of } \mathcal{A} \\
&\leq \frac{(1 + \epsilon')^3}{(1 - \epsilon')^2} \left(\sum_t \ell_f^t + 3B + \frac{A(d, T)}{\gamma \cdot \epsilon'} \right) && \text{by 3 applied to } f
\end{aligned}$$

The theorem then follows as $\frac{(1+\epsilon')^3}{(1-\epsilon')^2} \leq (1 - \epsilon)^{-1}$ for $\epsilon' = \epsilon/5$. ■

The small-loss guarantee without knowing α. So far, we presented the results in terms of approximate regret and assuming we have α, an upper bound for the maximum independent set, as an input. Next we show that we can use this algorithm with the classical doubling trick without knowing α, and achieving low regret both in expectation as well as with high probability, not only approximate regret. We start with a large ϵ and small α and halve and double them respectively, when observing that they are not set right. There are two issues worth mentioning.

First, observe that computing the maximum independent set is challenging since this task is NP-hard to approximate. However, if one looks carefully into our proofs, we just require knowledge of a maximal independent set on the γ-frozen arms and not one of maximum size. This can be easily computed greedily at each round and therefore our algorithm can handle changing graphs without requiring knowledge of the maximum independence number.

Second, unlike full feedback, partial feedback does not provide access to the loss of the comparator $L^\star$. As a result, we apply doubling trick on the loss of the algorithm instead and then bound the regret of the algorithm appropriately. Using the loss of the algorithm instead is called *self-confident approach* [13]. Combined with standard doubling trick arguments, this gives the following lemma whose proof is provided in Appendix A.2 for completeness.

Lemma 2.3. Suppose we have a randomized algorithm that takes as input any $\epsilon > 0$ and guarantees that, for some $q \geq 1$ and some function $\Psi(\cdot)$, and any $\delta > 0$, with probability $1 - \delta$, for any time horizon s and any comparator f:

$$(1-\epsilon)\sum_{t=1}^{s} \ell_{A(t)}^t \leq \sum_{t=1}^{s} \ell_f^t + \frac{\Psi(\delta)}{\epsilon^q}.$$

Assume using this algorithm over multiple phases (by restarting the algorithm when a phase ends). We run each phase τ with $\epsilon_\tau = 2^{-\tau}$ until $\epsilon_\tau \widehat{L_\tau} > \frac{\Psi(\delta)}{(\epsilon_\tau)^q}$ where $\widehat{L_\tau}$ denotes the cumulative loss of the algorithm for phase τ. For any $\delta > 0$, the regret for this multi-phase algorithm is, with probability at least $1-\delta$:

$$\textit{Regret} \quad \leq O\Big((L^\star)^{\frac{q}{q+1}} \Psi\Big(\tfrac{\delta}{\log(L^\star+1)+1}\Big)^{\frac{1}{q+1}} + \Psi\Big(\tfrac{\delta}{\log(L^\star+1)+1}\Big) + 1\Big)$$

Combining the two observations, we prove the following small-loss bound.

Theorem 2.3. Let $\mathcal{A}$ be any full information algorithm with ϵ-approximate regret bounded by $L \cdot A(d,T)/\epsilon$ when run on losses in $[0, L]$ and with parameter $\epsilon > 0$. If one runs the *Dual-Threshold Freezing Algorithm* (Algorithm 1) as in Theorem 2.2 and using the doubling scheme as in Lemma 2.3 and tuning α appropriately on each phase, then for any $\delta > 0$, with probability at least $(1-\delta)$ the regret of this algorithm is bounded by $\mathcal{O}\Big(\big((L^\star)^{2/3}(\alpha A(d,T))^{1/3} + \alpha A(d,T)\big)\log\Big(\frac{d\log(L^\star+1)}{\delta}\Big)\Big)$.

Proof. First for simplicity assume that α is known in advance. In this case, using Theorem 2.2, we can conclude that for any $\delta, > 0$, Algorithm 1 run with $\mathcal{A}$ enjoys an ϵ-approximate regret guarantee of $\mathcal{O}\Big(\frac{\alpha\cdot(A(d,T)+\log(d/\delta))}{\epsilon^2}\Big)$. Hence, running Algorithm 1 while tuning ϵ-parameter using doubling trick as in Lemma 2.3 with $\Psi(\delta) = \mathcal{O}(\alpha \cdot (A(d,T) + \log(d/\delta))$ and $q = 2$ yields the regret guarantee of

$$\mathcal{O}\left(\left((L^\star)^{2/3}(\alpha A(d,T))^{1/3} + \alpha A(d,T)\right)\log\left(\frac{d\log(L^\star+1)}{\delta}\right)\right)$$

If α is not known in advance, we can begin with a guess (say $\alpha' = 1$) and double the guess every time that this is incorrect, i.e. the maximal independent set of the γ-frozen nodes has more than α' nodes. We make at most $\log(\alpha)$ updates. Within one phase with the same update, the previous guarantee holds with probability at least some δ'. At the time of each update we can lose an extra of at most 1. For the rest of the rounds, the guarantees work additively. Therefore, setting $\delta' = \delta/\log(\alpha)$, we obtain the previous guarantee with an extra $\log(\alpha)$ decay in the guarantee. Since $\alpha < d$, the dependence on $\log(\alpha)$ is dropped in the O notation of the regret bound. ■

2.5 Optimal small-loss guarantees for bandit feedback

To achieve optimal dependence on $L^\star$, we need to better understand the places where the inefficiency arises. The first such place is when we apply the bound of the full-information which, in a black-box analysis, needs to have dependence both on the magnitude of losses, $L \leq 1/\gamma'$, and on the approximation parameter ϵ'. Instead of applying this bound, we provide a refined analysis that relates the expected estimated loss of the full information algorithm to the sum of the cumulative estimated losses of all the arms. Using multiplicative weights as a full-information algorithm guarantees that the cumulative estimated losses of all the arms are close to each other (Lemma 2.4) which enables us to remove this inefficiency. This was also used by Allenberg et al. [5] to prove optimal pseudo-regret guarantees but their analysis did not extend to high-probability. To derive the high-probability guarantee, we address the second inefficiency of the black-box, where to bound the negative bias of the comparator's cumulative estimated loss by its cumulative actual loss, we again had dependence on both the magnitude of the estimated losses and ϵ. For that we apply the implicit exploration idea of Kocák et al. [81] which creates a negative bias to all arms and not only the arms that are frozen (Lemma 2.5). Although Neu [103] used implicit exploration to provide high-probability uniform bounds his results did not extend to small-loss. Combining our framework with both multiplicative weights and implicit exploration, we obtain an algorithm we term *GREEN-IX* (Algorithm 2) that, with high-probability, guarantees regret bound of $O(\sqrt{L^\star})$.

Theorem 2.4. For any $\delta > 0$, GREEN-IX run with learning parameter $\epsilon' = \epsilon/2$ guarantees an ϵ-approximate regret of $O\Big(\frac{d\log(d/\delta)}{\epsilon}\Big)$ with probability at least $1-\delta$.

Lemma 2.4 (implied by the proof of Theorem 2 in [5])**.** When using multiplicative

```
Algorithm 2: GREEN-IX
Require: Number of arms $d$, learning parameter $\epsilon'$.
1: Initialize $\tilde{p}_a^1$ for arm $a$ ($\tilde{p}_a^t = 1/d$), their cumulative losses ($\tilde{L}_a^0 = 0$). Set $t = 1$.
2: for $t = 1$ to $T$ do
3:     Freeze arm $a$ if its probability $\tilde{p}_a^t$ is below threshold $\gamma = \epsilon'/d$ to create the set $F^t = \{i : p_a^t < \gamma\}$.
4:     Normalize probabilities of unfrozen arms so that they form a distribution:
           $p_a^t = 0$ if $a \in F^t$ and $p_a^t = \frac{\tilde{p}_a^t}{1-\sum_{a' \in F^t} \tilde{p}_{a'}^t}$ otherwise.
5:     Draw arm $A(t) \sim p^t$ and incur loss $\ell_{A(t)}^t$.
6:     Compute biased estimate of losses via implicit exploration $\zeta = \epsilon'/(2d)$:
           $\tilde{\ell}_a^t = \frac{\ell_a^t}{p_a^t+\zeta}$ if $a = A(t)$ and $\tilde{\ell}_a^t = 0$ otherwise.
       and update the cumulative losses of all arms $\tilde{L}_a^t \rightarrow= \tilde{L}_a^{t-1} + \tilde{\ell}_a^t$.
7:     Update $\tilde{p}_a^{t+1}$ via multiplicative weights with learning rate $\eta = \epsilon'/(2d)$:
           $\tilde{p}_a^{t+1} = \frac{\exp(-\eta \tilde{L}_a^t)}{\sum_{a'} \exp(-\eta \tilde{L}_{a'}^t)}$.
8: end for
```

weights as the full-information algorithm, for any two arms a and a',

$$\sum_{t=1}^{T} \tilde{\ell}_a^t \leq \sum_{t=1}^{T} \tilde{\ell}_{a'}^t + \frac{1}{\gamma} + \frac{\ln(1/\gamma)}{\eta}$$

Proof. Let T_a be the last round that a is not frozen. Thus its probability (before normalization) is then greater than γ.

$$\gamma \leq \tilde{p}_a^{T_a} = \frac{\exp\left(-\eta \sum_{t=1}^{T_a-1} \tilde{\ell}_a^t\right)}{\sum_j \exp\left(-\eta \sum_{t=1}^{T_j-1} \tilde{\ell}_j^t\right)} \leq \frac{\exp\left(-\eta \sum_{t=1}^{T_a-1} \tilde{\ell}_a^t\right)}{\exp\left(-\eta \sum_{t=1}^{T_{a'}-1} \tilde{\ell}_{a'}^t\right)}$$

As a result:

$$\sum_{t=1}^{T_a-1} \tilde{\ell}_a^t \leq \sum_{t=1}^{T_{a'}-1} \tilde{\ell}_{a'}^t + \frac{\ln(1/\gamma)}{\eta} \Rightarrow \sum_{t=1}^{T} \widetilde{\ell_a^t} \leq \sum_{t=1}^{T} \tilde{\ell}_{a'}^t + \frac{1}{\gamma} + \frac{\ln(1/\gamma)}{\eta},$$

where the last inequality follows as $\tilde{\ell}_a^t \leq 1/\gamma$ for all arms at all times and the estimated loss of a is 0 after round T_a by definition of T_a. ■

Lemma 2.5 (implied by Corollary 1 in [103]). For any $\delta > 0$, with probability at least $1 - \delta$, any full information algorithm run on estimated losses $\tilde{\ell}^t$ with implicit exploration satisfies for all arms $a \in [d]$ simultaneously:

$$\sum_{t=1}^{T}\left(\tilde{\ell}_a^t - \ell_a^t\right) \leq \frac{\log(d/\delta)}{2\zeta}$$

Proof. The lemma essentially follows from Corollary 1 in [103], that proves the analogous statement when there is just implicit exploration without freezing. Let's consider some fictitious losses $\bar{\ell}_a^t$ that are equal to the actual losses for all arms $a \notin F^t$ and 0 for arms $a \in F^t$ and let $\hat{\ell}_a^t$ be the estimated loss with just implicit exploration the losses $\bar{\ell}_a^t$. Then Corollary 1 in [103] establishes that: $\sum_{t=1}^{T}\left(\hat{\ell}_a^t - \bar{\ell}_a^t\right) \leq \frac{\log(d/\delta)}{2\zeta}$ simultaneously for all a with probability at least $1 - \delta$. The lemma follows by noting that the fictitious estimated losses are equal to the true estimated losses, i.e. $\hat{\ell}_a^t = \tilde{\ell}_a^t$, since all the non-frozen arms have the same actual losses and that the fictitious actual losses are no greater than the true actual losses, i.e. $\bar{\ell}_a^t \leq \ell_a^t$ since the only difference occurs on arms with $\bar{\ell}_a^t = 0$ and all the actual losses are non-negative. ■

Lemma 2.6 (see for instance [35]). Multiplicative weights with learning rate η applied on the estimated losses satisfies:

$$\sum_t \sum_a \tilde{p}_a^t \tilde{\ell}_a^t - \sum_t \tilde{\ell}_f^t \leq \eta \sum_t \sum_a \tilde{p}_a^t \left(\tilde{\ell}_a^t\right)^2 + \frac{\log(d)}{\eta}$$

Proof of Theorem 2.4. The proof follows the roadmap of the proof of Theorem 2.2 but handles the suboptimal places of the black-box theorem's proof by applying Lemmas 2.4 and 2.5. We show that for each arm f, the guarantee holds with failure probability $\delta' = \delta/d$. Therefore the guarantee holds against all the arms f simultaneously with probability at least $1-\delta$. More formally:

$$\begin{aligned}
(1-\epsilon)\sum_t \ell^t_{A(t)} &= (1-\epsilon)\sum_t\sum_i (p^t_a+\zeta)\cdot\tilde{\ell}^t_a && \text{by definition of } \widetilde{\ell^t_i}\\
&\le \frac{1-\epsilon}{1-\epsilon'}\sum_t\sum_a \tilde{p}^t_a\tilde{\ell}^t_a + \zeta\sum_t\sum_a\tilde{\ell}^t_a && \text{by Lemma 2.1}\\
&\le \frac{1-\epsilon}{1-\epsilon'}\sum_t \tilde{\ell}^t_f + \eta\sum_t\sum_a \tilde{p}^t_a\left(\tilde{\ell}^t_a\right)^2\\
&\quad + \frac{\log(d)}{\eta} + \zeta\sum_t\sum_a\tilde{\ell}^t_a && \text{by Lemma 2.6}\\
&\le \frac{1-\epsilon}{1-\epsilon'}\sum_t \tilde{\ell}^t_f + (\eta+\zeta)\sum_t\sum_a\tilde{\ell}^t_a + \frac{\log(d)}{\eta} && \text{as } \ell^t_a\le 1 \text{ and } \tilde{p}^t_a \le p^t_a+\zeta\\
&\le \frac{1-\epsilon}{1-\epsilon'}\sum_t \tilde{\ell}^t_f + (\eta+\zeta)\sum_{t=1} d\tilde{\ell}^t_f\\
&\quad + d(\eta+\zeta)\left(\frac{1}{\gamma}+\frac{\ln(1/\gamma)}{\eta}\right) + \frac{\log(d)}{\eta} && \text{by Lemma 2.4}
\end{aligned}$$

Now we use the strict negative bias of Lemma 2.5 to get that with probability at least $(1-\delta')$ we can continue the above inequalities as:

$$\begin{aligned}
(1-\epsilon)\sum_t \ell^t_{A(t)} &\le \frac{1-\epsilon}{1-\epsilon'}\sum_t \ell^t_f + \frac{\log(d/\delta')}{2\zeta} + (\eta+\zeta)\sum_{t=1} d\ell^t_f\\
&\quad + d(\eta+\zeta)\left(\frac{1}{\gamma}+\frac{\ln(1/\gamma)}{\eta}+\frac{\log(d/\delta')}{2\zeta}\right) + \frac{\log(d)}{\eta}\\
&\le \sum_t \ell^t_f + \frac{4d\log(d)+2\log(d^2/\delta)}{\epsilon} + d\Big(1+2\ln(2/\epsilon)+\log(d^2/\delta)\Big).
\end{aligned}$$

where the final inequality is derived by replacing the parameters γ, ζ, η, and δ', and using the fact that $\frac{1-\epsilon}{1-\gamma d} + (\eta+\zeta)d \le 1$ for the selection of the parameters. ■

Corollary 2.1. GREEN-IX applied with doubling trick on parameter ϵ guarantees regret of $\widetilde{O}\Big(\sqrt{d\log(d/\delta)\cdot L^\star} + \log(d/\delta)\Big)$ with probability at least $1-\delta$, and hence expected regret at most $\widetilde{O}(\sqrt{d\log(d)\cdot L^\star} + \log(d/\delta))$.

The proof follows similarly to the one of Theorem 2.2 by applying Lemma 2.3 with $\Psi(\delta) = O(d \log(d/\delta))$ and $q = 1$.

2.6 Remarks

More information about the paper. The results presented in this chapter are joint work with Karthik Sridharan and Éva Tardos [93]. Our reduction can capture more important partial feedback paradigms such as combinatorial semi-bandits, contextual bandits, and bandits with dynamically evolving comparators. In particular, one important additional result in the paper is about combinatorial semi-bandits where we provide optimal high-probability small-loss guarantees similar to the results in Section 2.5. The latter result stems from combining our black-box reduction with a) again the implicit exploration of Kocák et al. [81], b) a truncated version of follow the perturbed leader of Neu [104], and c) the geometric resampling idea of Neu and Bartok [105]. Another interesting additional result is that, for the case where the feedback graph is fixed across time, the analysis provided in Section 2.5 can be extended to provide guarantees with optimal dependence on $L^\star$ that depend on the minimum clique partition, instead of the number of arms. Extending this result to evolving graphs and replacing the clique partition by the maximum independence number is a major open question coming out of our work.

Other small-loss bounds with partial feedback. The work described in this section is not the first to provide small-loss bounds with partial feedback, however these guarantees are generally challenging when moving away from full feedback, e.g. see the open problem in [3]. For example, such results existed for the pure bandit feedback setting: e.g., the paper by Allenberg et al. [5] which we extended here, as well as my previous joint work with Dylan Foster, Zhiyuan Li, Karthik Sridharan, and Éva Tardos on online mirror descent with log-barrier regularizer [53]. Other settings where such small-loss guarantees were analyzed are the label-efficient prediction setting [39], the combinatorial semi-bandit setting [104] which we also extend in our work as described in the previous paragraph, and (subsequently to our work) contextual bandits [33]. All these results rely on algorithms tailored to the setting and give guarantees that only hold in expectation for the weaker notion of pseudo-regret that compares to an arm fixed in advance (not the best in hindsight). In contrast, our guarantees are for general graph-based feedback, hold with high probability against the ex post optimal arm, and our reduction for the suboptimal rate of $(L^\star)^{2/3}$ is black-box.

Other data-dependent guarantees. Small-loss guarantees are a particular form of data-dependent bounds. These guarantees improve on the worst-case when the problem has a nice structure but do not rely on this structure to be perfectly present and the guarantees gracefully degrade as we deviate from it. The nice structure in our case is the fact that there exists an action that has small aggregate loss; whenever this happens, we see significantly better performance. Another important structure that has been employed to provide robust data-dependent guarantees for adversarial online learning are guarantees that become better if the variance between the losses in the realized sample path is small [67, 127]. In the next chapter, we will also discuss an orthogonal version of data-dependent guarantees that, instead of utilizing a nice structure in adversarial online learning, aim to make algorithms that rely on a particular assumption (such as data being i.i.d.) robust to this assumption not completely holding.

Beyond the classical regret notion. Although we mostly focus on regret bounds with respect to the classical benchmark that compares to the loss of the best fixed action in hindsight, our results extend seamlessly to the stronger benchmark of shifting regret [69] that compares to a sequence of comparator arms (not changing too often). This extension is important when we discuss efficiency of learning outcomes in dynamic environments in Chapter 6. We should mention that there are a variety of other benchmarks considered in adversarial online learning: for example, sleeping regret [26, 55, 30] allows different arms to only be available in particular rounds and adaptive regret [91, 43] requires to have good regret for each time interval; we will come back to those in Chapter 6.

Partial-feedback settings outside of graph-based feedback. In this chapter, we focused on graph-based feedback, a combinatorial feedback structure where the losses of different elements are observed separately. Apart from providing a clean model to obtain intuition on handling side-information, this model captures important partial feedback settings as special cases as discussed. However, there are settings not captured in our framework. The most notable examples are feedback settings that are more continuous such as linear bandits [42, 118] and metric bandits [80]. Another such feedback setting that is relevant for packet routing is end-to-end routing [15] where the learner receives feedback for the whole path and not for all the segments as in combinatorial semi-bandits.

CHAPTER 3

ROBUSTNESS TO ADVERSARIAL CORRUPTIONS

The explore-exploit trade-off in online decision-making is often more evident in settings where the performance of the alternatives has some natural stochasticity. Unlike adversarial online learning which stems from game-theoretic considerations, the origin of sequential learning when the input is i.i.d. lies in the design of sequential experiments [113]. Moving beyond the classical statistical approach where all experiments were initially conducted before analyzing the data, sequential experiment design allowed to adaptively balance the exploration needed to identify the most profitable alternative with exploiting it once it is identified. This gave rise to the stochastic multi-armed bandit setting [11] which nicely encapsulates this explore-exploit trade-off.

The stochastic multi-armed bandit setting has become more useful due to the rise of online marketplaces which comes with many opportunities for higher adaptivity during sequential decision-making. Consider online advertising as an example where a platform like Google or Facebook needs to select which ad to display at a particular pageview. Different ads have different propensity to be clicked; this is expressed by the click-through-rate (probability of a click if the ad is displayed). The platform aims to select ads with high click-through rate in order to provide relevant content to the user and revenue to itself (as it is typically paid per-click). As a result, the crucial task in such an application is to identify which ad is the most profitable while also making sure that suboptimal actions are not selected too often. This trade-off also appears in recommender systems where the different alternatives may correspond to restaurants and the reward from an action could be associated with the experience of the user. This again is related to a distribution capturing the inherent quality of the restaurants.

In this chapter, we discuss a major roadblock in practically employing stochastic multi-armed bandit algorithms: the input is not completely i.i.d. but may be subject to the existence of fraudulent data. This is a prominent issue in the applications of interest. For example, in online advertising, there exists the phenomenon of click fraud. In one instantiation of click fraud, an attacking advertiser may try to obtain impressions from a competing ad and deliberatly not click it. This way the attacking advertiser may manipulate the platform towards thinking that the latter ad has a low click-throuh-rate and therefore should not be often selected. Altough platforms actively spend resources to identify when click fraud occurs, it is unreasonable to assume that it will be completely eradicated. Similarly, in recommender systems, there are often fake reviews; again platforms try to penalize such corrupted activity but again one cannot hope that all of it will be completely eliminated. The challenge we face is that classical algorithms completely fail even with a very small amount of fraudulent activity as we will see in Section 3.3. Addressing this challenge, we will suggest a way to make stochastic multi-armed bandit algorithms robust to such corruptions in the data.

3.1 Preliminaries on stochastic multi-armed bandit learning

Stochastic multi-armed bandits. The framework for stochastic multi-armed bandits is similar to the one of adversarial online learning described in Section 2.1. It differs from it in that the losses or rewards are drawn from distributions fixed in advance instead of being adversarially selected. Since this is more common in our applications, we switch the presentation to be about rewards in this chapter.

More formally, the decision-maker or learner has again access to a set of k alternatives that we will refer to as *arms* or *actions*. Each arm $a \in \{1, \dots, k\}$ is associated with a distribution $\mathcal{F}(a)$ with mean $\mu(a)$. The distributions are assumed to have positive measure only on rewards in $[0, 1]$ and are unknown to the learner. At round $t = 1, \dots, T$, the following process occurs:

1. The learner selects a probability distribution $p^t \in \Delta(k)$ over the k possible arms, i.e. $\sum_{a=1}^{k} p_a^t = 1$.
2. For each arm a, a reward $r_a^t \sim \mathcal{F}(a)$ is drawn from the corresponding distribution where $r_a^t \in [0, 1]$ is assumed to lie in $[0, 1]$.
3. The learner then draws action $A(t) \sim p^t$ from the distribution p^t she committed to and gains the reward of the selected arm $r_{A(t)}^t$.
4. The learner observes the reward $r_{A(t)}^t$ only for the selected arm $A(t)$.

Regret notions. If the learner knew in advance the distributions $\mathcal{F}(a)$, she would always select the arm $a^\star = \arg\max_a \mu(a)$ as it provides the highest expected reward. However, this information is not known in advance. As a result, the notion of performane in stochastic bandits captures how costly this lack of distributional information ends up being for the algorithm. More formally, the notion of pseudoregret corresponds to the difference between the regret obtained by the algorithm and the reward of arm $a^\star$.

$$PseudoReg = \mathbb{E}\left[\sum_t \left[r_{a^\star}^t - r_{A(t)}^t\right]\right] = \sum_t \mathbb{E}[\mu(a^\star) - \mu(A(t))].$$

A stronger regret notion is that of *actual regret* that compares the realized performance of the algorithm to the realized performance of the best arm in hindsight instead of expectation against $a^\star$; note that the best arm in hindsight may be different than the ex-ante optimal arm $a^\star$. More formally:

$$Regret = \max_a \sum_t \left[r^t_{A(t)} - r^t_a \right]$$

The actual regret is a random variable that depends on the random rewards, the randomness used by the learner, and the randomness of the adversary. We say that a regret bound $R(T, \delta)$ *holds with probability* $1 - \delta$ if $\mathbb{P}[Regret < R(T, \delta)] > 1 - \delta$ where the probability is taken over all the three sources of randomness described. Note that by Jensen's inequality, $PseudoReg \leq \mathbb{E}[Regret]$. We can often obtain improved bounds for pseudoregret since it allows us to offset large positive regret events with large negative regret events (see discussion in Section 3.6).

Classical guarantees. One can generally exploit the stochasticity in the input to obtain improved guarantees compared the adversarial online learning guarantees of $\sqrt{T \log(k)}$ which we discussed in the previous chapter. The property that the input is stochastic is more useful when arm $a^\star$ is more easily identifiable. As a result, the guarantees tend to scale inversely with the so-called gaps of the arms a, i.e. $\Delta(a) = \mu(a^\star) - \mu(a)$, which captures how easily identifiable $a^\star$ is.[1] More concretely, the guarantees are of the form $\Theta\Big(\sum_{a \neq a^\star} \frac{\log(kT/\delta)}{\Delta(a)}\Big)$ for actual regret with probability at least $1 - \delta$, and $\Theta\Big(\sum_{a \neq a^\star} \frac{\log(kT)}{\Delta(a)}\Big)$ for pseudoregret. In Section 3.3, we show how these bounds are obtained for a classical stochastic bandit algorithm and then illustrate why such algorithms are not robust to corruptions in the data.

The above guarantees may seem meaningless when there are arms with $\Delta(a) \leq 1/\sqrt{T}$. For those summands, the inverse dependence on the gap may

[1]We note that $a^\star$ is one arm with optimal mean and this does not preclude the existence of other arms with the same mean. If more than one such arms exist, let $a^\star$ be an arbitrary arm with optimal mean and the other arms $a \neq a^\star$ with optimal mean have gap $\Delta(a) = 0$. To simplify presentation, we assume that $a^\star$ is the unique arm with highest mean.

initially seem vacuous; for instance, when there are two optimal arms $a, a^\star$ with the same mean, the upper bound becomes infinite as $\Delta(a) = 0$. However, the inverse dependence on the gap can be replaced by $\Delta(a) \cdot T$ in the case of pseudo-regret and $\sqrt{T}$ in the case of actual regret.[2] For simplicity of exposition, we omit this from the remaining of the discussion.

3.2 Stochastic bandits with adversarial corruptions

Corrupted model. We now slightly modify the stochastic bandit learning setting described in the previous section to incorporate adversarial corruptions in the data. We consider an adversary who can corrupt some of the stochastic rewards. The adversary is adaptive in the sense that the corrupted rewards can be a function of the realization of the stochastic rewards up to that point and of the learner's choices in the previous rounds. More formally, at round $t = 1, \dots, T$:

1. The learner selects a probability distribution $p^t \in \Delta(k)$ over the k possible arms, i.e. $\sum_{a=1}^{k} p_a^t = 1$.
2. For each arm a, a reward $r_a^t \sim \mathcal{F}(a)$ is drawn from the corresponding distribution where $r_a^t \in [0, 1]$ is assumed to lie in $[0, 1]$.
3. The adversary observes the realizations of r_a^t as well as the learner's choices p_a^t and returns corrupted feedback $\tilde{r}_a^t$ for all arms a.
4. The learner then draws action $A(t) \sim p^t$ from the distribution p^t she committed to and gains the reward of the selected arm $r_{A(t)}^t$.
5. The learner observes the corrupted feedback $\tilde{r}_{A(t)}^t$ only for selected arm $A(t)$.

[2]If two arms have the same mean, then concentrantion bounds can only establish that the reward of the algorithm will be at most $\sqrt{T}$ worse than the ex-post best arm with high-probability.

Delving into the model, steps 1, 2, and 4 are as in the stochastic bandit model. The only modification is that there exists an extra step 3 in which the adversary alters the feedback which the learner observes for the selected arm in step 5.

Note that, in the described model, we assume that the reward earned is the one before corruption and the adversary only corrupts the feedback received. This makes sense in settings with fraudulent activity such as fake reviews; the corruption in a review does not improve the user experience (earned reward). Our algorithm extends to the setting where both the reward earned and the feedback received are corrupted; we contrast the two settings further in Section 3.6.

Desiderata. We aim for guarantees that gracefully degrade based on how corrupted the setting is. More formally, we quantify the amount of corruption injected at round t by the maximum difference the adversary injected in any arm: $\max_a |r_a^t - \tilde{r}_a^t|$. An instance is *C-corrupted* if the total injected corruption (across time) is at most

$$\sum_t \max_a |r_a^t - \tilde{r}_a^t| \leq C$$

for all realizations of the random variables. The adversary is assumed to be adaptive, in the sense that she has access to all the realizations of random variables for all rounds $\tau < t$ and the realization of rewards at round t but only knows the player's distribution at round t and not the arm a^t.

We aim for algorithms with the following three properties:

1. **Stochastic:** Retain the stochastic bandit guarantee when $C = 0$.
2. **Robust:** Have the guarantee degrade gracefully as a function of C.
3. **Agnostic:** Do not assume knowledge of parameter C.

Note that a linear degradation with respect to C is unavoidable for the robustness property. Consider two arms a and a' with means $\mu(a) = 1$ and $\mu(a') = 0$ respectively and an adversary who returns $\tilde{r}^t(a) = \tilde{r}^t(a') = 1$ for both arms in the first C rounds. No algorithm can do better than selecting uniformly at random among these arms for the first C rounds, which leads to a pseudoregret of $C/2$.

3.3 Click fraud attack against classical bandit algorithms

Active Arm Elimination algorithm. The starting point of our design is the *Active Arm Elimination* algorithm [51], which provides a simplified analysis of the stochastic bandit guarantee compared to the more famous UCB algorithm [11].

This algorithm is based on the following idea: in an initial *exploration phase*, we pull arms in a round-robin fashion and compute an estimate $\tilde{\mu}(a)$ as the average empirical reward of arm a (average reward of $a^\star$ when observed). After $n(a)$ pulls of arm a, usual concentration bounds establish that with probability at least $1 - 1/T$, the difference of the empirical and actual means is at most $\mathsf{wd}(a) = \sqrt{\log(T)/n(a)}$. We say that $[\tilde{\mu}(a) - \mathsf{wd}(a), \tilde{\mu}(a) + \mathsf{wd}(a)]$ is the confidence interval of arm a.

If, at some point, the difference between the empirical means of two arms a and a' becomes larger than the widths of the confidence intervals, i.e., $\tilde{\mu}(a') - \tilde{\mu}(a) > \mathsf{wd}(a) + \mathsf{wd}(a')$, then with high probability arm a is not the optimal arm. Once this happens, the algorithm eliminates arm a by removing it from the round-robin rotation. After both arm a and the best arm $a^\star$ are pulled $O\left(\frac{\log(T)}{\Delta(a)^2}\right)$ times, the confidence intervals will be small enough that arm a will be eliminated.

Eventually all arms but the optimal are eliminated and we enter what is called the *exploitation phase*. In this phase we only pull the arm $a^\star$ which has the highest mean. Before we enter exploitation, each suboptimal arm a is pulled at most $O(\frac{\log(T)}{\Delta(a)^2})$ times. Each of those suboptimal pulls incurs regret $\Delta(a)$ in expectation which leads to the pseudo-regret bound of $O(\sum_{a \neq a^\star} \frac{\log(T)}{\Delta(a)})$. This bound can also be converted to a high probability bound for any given failure probability $\delta > 0$ if we replace $\log(T)$ by $\log(kT/\delta)$.

Click fraud attack. The above algorithm is not robust to corruptions as an adversary can easily target the exploration phase, manipulating the algorithm to believe that $a^\star$ is not the optimal arm.

Consider the case where we only have three arms a and $a^\star$ with $\mu(a) = 0.9$, $\mu(a') = 0$ and $\mu(a^\star) = 1$. As illustrated in the previous exposition, the algorithm will realize that arm a' is really suboptimal after a logarithmic number of rounds and subsequently remove it from the round-robin rotation. As a result, what the adversary can do is to make the optimal arm $a^\star$ look exactly as a' to provoke its elimination. This can be easily achieved by modifying the feedback of arm $a^\star$ to 0 for a logarithmic number of rounds. As a result, $a^\star$ is then indistinguishable from arm a' and will fast get eliminated for the same reason. Subsequently, we incur a regret equal to the gap $\Delta(a) = 0.1$ for every single round, leading to a linear regret which violates the robustness property as $C = O(\log T)$ in the setting.

This is exactly the click fraud attack that occurs in online advertising. The advertiser that has ad a may try to get impressions of the optimal ad $a^\star$ to manipulate the platform to believe that $a^\star$ is really not effective and have their own ad being selected for the remainder of the time.

3.4 Warm-up: c-corrupted setting with valid upper bound c

To approach the setting, we first simplify it by assuming that we have access to an upper bound c on the corruption, i.e. $C < c$, and aiming for guarantees gracefully degrading with c. This is not an assumption we wish to make in practice for various reasons. First, the typical way that such an upper bound may arise is via setting a pretty loose upper bound to be sure it is a valid upper bound; this will result in violating the stochastic property as, even if the setting ending up being uncorrupted ($C = 0$), we will end up scaling with the loose upper bound c. Even more damagingly, from a game-theoretic viewpoint, if the alogirhtm has a hard-coded bound c it is robust to, the adversary needs to just try to add a little more corruption since the algorithm then only satisfies the robustness property for $C < c$. As a result, it is crucial to be agnostic to the amount of corruption. However, this setting will serve as a useful building block in our framework.

Enlarged confidence intervals. With such an upper bound c, there is a simple modification of Active Arm Elimination that gracefully degrades with c. In particular, we can enlarge the confidence intervals to account for this quantity. More formally, setting $\mathsf{wd}(a,t) = \sqrt{\frac{\log(T)}{n(a,t)}} + \frac{c}{n(a,t)}$ where $n(a,t)$ is the number of times the arm has been played until time t, the resulting algorithm has performance $O\Big(\sum_{a \neq a^\star}\Big(\frac{\log(kT/\delta)+c}{\Delta(a)}\Big)\Big)$ with probability $1-\delta$ if the bound c is indeed a valid upper bound. This comes from two lemmas that will be useful moving forward; their proofs follow standard stochastic bandit arguments and are provided in Appendix B.1 for completeness.

Lemma 3.1. Assume that c is a valid upper bound for the total corruption and we run active arm elimination with $\mathsf{wd}(a,t) = \sqrt{\frac{\log(2kT/\delta)}{n(a,t)}} + \frac{c}{n(a,t)}$. Then, with probability at least $1-\delta$, arm $a^\star$ never becomes eliminated.

Lemma 3.2. Assume that c is a valid upper bound for the total corruption and we run Active Arm Elimination with $\mathsf{wd}(a,t) = \sqrt{\frac{\log(2kT/\delta)}{n(a,t)}} + \frac{c}{n(a,t)}$. Then, with probability at least $1-\delta$, all arms $a \neq a^\star$ become eliminated after $N(a) = \frac{36\log(2kT/\delta)+6c}{\Delta(a)^2}$ plays.

3.5 Main result: Multi-layer Active Arm Elimination

We are now ready to provide our approach that combines the stochastic, robustness, and agnostic properties. To make the presentation more modular, we first describe the simpler setting where we wish to obtain the usual stochastic bound of $O\Big(\sum_{a\neq a^\star}\frac{\log(kT/\delta)}{\Delta(a)}\Big)$ if the input is purely stochastic while simultaneously guaranteeing $O\Big(k\cdot c\cdot\sum_{a\neq a^\star}\frac{\log(kT/\delta)^2}{\Delta(a)}\Big)$ if the input is C-corrupted with corruption level $C \leq c$ upper bounded by a known c. Subsequently, we will extend it to the agnostic case where we will provide the same result with respect to the realized C without assuming knowledge of any upper bound.

Stochastic or corrupted with known valid upper bound c

To deal with this double purpose (stochastic and c-corrupted when $C \leq c$ for a known c), we run in parallel two instances each targeting one of the two goals; at each round we select the instances with appropriate probabilities described below. Intuitively, the first instance is selected more often and quickly identifies the best arm if the input is stochastic, but is not robust to corruptions. The second instance is slower but more precise, in the sense that it can tolerate corruptions.

Since the second instance is more trustworthy, if the second instance decides to eliminate a certain arm a, we eliminate the same arm in the faster instance.

Decrease experienced corruption by sub-sampling. To keep the regret low if the input is stochastic, the second instance of active arm elimination cannot pull a suboptimal arm too many times, therefore enlarging the confidence intervals by c is not effective. The *main idea* of the algorithm is to make arm a behave as if it was almost stochastic in the slower instance even when there exists corruption $C \leq c$ in the data, by running the slower instance with low probability. If the learner selects the slower instance with probability $1/c$ then, when the adversary adds a certain amount of corruption at some round, the slower instance observes that corruption with probability $1/c$. Hence, the expected amount of corruption the learner observes in the slower instance is $C \cdot \frac{1}{c}$ which is less than 1 in expectation (as $C \leq c$) and less than $c^{\mathsf{S}} = \log(2kT/\delta)$ with high probability (at least $1 - \delta$).[3] This makes the arms behave almost like stochastic arms in the slower instance despite the potential existence of corruption C via only enlarging the confidence intervals by c^{S}. The slower instance thus becomes robust to corruption by randomly sparsifying the corruption it experiences.

Fast-slow active arm elimination race. We obtain our algorithm by combining this *random sparsification* idea with enlarging confidence intervals. We have two instances of active arm elimination which we denote by F (fast) and S (slow). Each instance keeps, for each arm a and time t an estimate of the mean $\tilde{\mu}^{\mathsf{F}}(a,t)$ and $\tilde{\mu}^{\mathsf{S}}(a,t)$ corresponding to the average empirical reward of that arm. It also keeps track of how many times each arm has been pulled in that instance

[3]The dependence on k and T since we need high-probability guarantees for every arm and at every time; hence the failure probability needs to allow for a union bound across all bad events.

$n^{\mathsf{F}}(a,t)$ and $n^{\mathsf{S}}(a,t)$. To handle the robustness to different levels of corruption, we enlarge the confidence intervals similarly to the previous section with $c^{\mathsf{F}} = 0$ and $c^{\mathsf{S}} = \log(8kT/\delta) + 3$ respectively. This allows us to define a notion of enlarged confidence interval in each of the instances as in the previous section with respective widths $\mathsf{wd}^{\ell}(a) = \sqrt{\frac{\log(8kT/\delta)}{n^{\ell}(a)}} + \frac{c^{\ell}}{n^{\ell}(a)}$ for $\ell = \{\mathsf{F},\mathsf{S}\}$. Also, each instance keeps a set of eliminated arms for that instance $\mathcal{I}^{\ell}$.

In each round, with probability $1 - 1/c$ we make a move in the fast instance: we choose the next active arm a in its round robin order, i.e., arm $a \in [k] \setminus \mathcal{I}^{\mathsf{F}}$ which was played less often, pull this arm and increase $n^{\mathsf{F}}(a)$ and update $\tilde{\mu}^{\mathsf{F}}(a)$ accordingly based on the (potentially) corrupted feedback. As usual, if there are two active arms a and a' such that $\tilde{\mu}^{\mathsf{F}}(a) - \tilde{\mu}^{\mathsf{F}}(a') > \mathsf{wd}^{\mathsf{F}}(a) + \mathsf{wd}^{F}(a')$ we eliminate a' by adding it to $\mathcal{I}^{\mathsf{F}}$.

With the remaining probability we make a move in the slow instance by executing the exact same procedure as described for the other instance. There is only one difference (which causes the two instances to be coupled): when we eliminate an arm a in S we also eliminate it in F.

This probabilistic selection of the instance leaves us with a potential problem: it is possible that all arms in the F instance end up being eliminated. If we reach that point, we play an arbitrary active arm of the slow instance, i.e., any arm $a \in [k] \setminus \mathcal{I}^{\mathsf{S}}$, without updating anything. Via this, we ensure that, when layer F failed to find the optimal arm, we select arms that are currently still active in the more robust slow instance and therefore we can still bound the total regret each arm causes by the number of rounds it survives in the slower instance. Crucially we do not update the estimates in this case as we have not subsampled the corruption in these samples and therefore they are less reliable.

The resulting algorithm is formally provided in Algorithm 3; to simplify notation there, we denote the selected arm as a^t instead of $A(t)$ and omit the dependence on t from empirical mean $\tilde{\mu}^\ell(a,t)$ and number of trials $n^\ell(a,t)$.

Algorithm 3: Fast-Slow Active Arm Elimination

```
Require: Number of arms $k$, horizon $T$, valid upper bound $c$ on corruption.
1: Initialize $n^\ell(a) = 0, \widetilde{\mu}^\ell(a) = 0, \mathcal{I}^\ell = \emptyset$ for all $a \in [k]$ and $\ell \in \{\mathsf{F}, \mathsf{S}\}$
2: For rounds $t = 1, \ldots, T$
3:     Sample algorithm $\ell$: $\ell = \mathsf{S}$ with probability $1/c$. Else $\ell = \mathsf{F}$.
4:     If $[k] \setminus \mathcal{I}^\ell \neq \emptyset$
5:         Play arm $a^t \leftarrow \arg\min_{a \in [k] \setminus \mathcal{I}^\ell} n^\ell(a)$
6:         Update $\tilde{\mu}^\ell(a^t) \leftarrow [n^\ell(a^t) \cdot \tilde{\mu}^\ell(a^t) + \tilde{r}^t_{a^t}]/[n^\ell(a^t) + 1]$ and $n^\ell(a^t) \leftarrow n^\ell(a^t) + 1$
7:         While exists arms $a, a' \in [k] \setminus \mathcal{I}^\ell$ with $\tilde{\mu}^\ell(a) - \tilde{\mu}^\ell(a') > \mathsf{wd}^\ell(a) + \mathsf{wd}^\ell(a')$
8:             Eliminate $a'$ by adding it to $\mathcal{I}^\ell$
9:             If $\ell = \mathsf{S}$ then eliminate $a'$ from fast algorithm by adding it to $\mathcal{I}^{\mathsf{F}}$
10:    Else
11:        Play an arbitrary arm in the set $[k] \setminus \mathcal{I}^{\mathsf{S}}$ without updating any estimate.
```

Towards the performance guarantee, Lemma 3.3 bounds the amount of corruption that actually enters the slow active arm elimination algorithm, which enables the regret guarantee in Theorem 3.1.

Lemma 3.3. If the total corruption is $C \leq c$ then the slow active arm elimination algorithm S observes, with probability at least $1 - \delta$, corruption of at most $\ln(1/\delta) + 3$ during its exploration phase (when $\ell = \mathsf{S}$).

Proof sketch. If one cared just about the expected corruption experienced when

$\ell = \mathsf{S}$, it is at most a constant number since the total corruption is at most C and it affects S with probability $1/c$. To prove a high-probability guarantee we require a concentration inequality on martingale differences (since the corruptions can be adaptively selected by the adversary). Since this makes the arguments notationally heavier, we provide the proof details in Appendix B.2. ■

Theorem 3.1. With probability $1 - \delta$, the fast-slow active arm elimination has regret $O\Big(\sum_{a \neq a^\star} \frac{\log(kT/\delta)}{\Delta(a)}\Big)$ for the stochastic case and $O\Big(k \cdot c \cdot \sum_{a \neq a^\star} \frac{(\log(kT/\delta))^2}{\Delta(a)}\Big)$ for the C-corrupted case with $C \leq c$.

Proof sketch. The result for the stochastic case follows standard arguments for stochastic algorithms (since we obtain double the regret of this setting as we run two such algorithms with essentially the same confidence intervals). For the C-corrupted case, we establish via Lemma 3.3 an upper bound on the corruption that will affect the slow active arm elimination algorithm S. Thanks to the subsampling, this upper bound is close to a constant instead of depending on the upper bound c which allows to not incur dependence on c in the stochastic case.

Having this upper bound, we can utilize Lemma 3.2 to obtain an upper bound on the number of plays of suboptimal arms in S. Since the algorithms are coupled, such a bound implies an upper bound on the regret that it can cause in F as well. This is because in expectation the arm is played at most $K \cdot C$ times more in F as it may be selected every single time in F prior to getting eliminated by S and F is selected c times more often than S. To obtain the above guarantee with high probability, we lose an extra logarithmic factor. The latter requires upper bounding with high probability the number of times between two consecutive times that $\ell = \mathsf{S}$. This comes via analyzing the first time that a p-biased coin with $p = 1/c$ returns heads; the details of the proofs are provided in Appendix B.2. ■

The general case

Multiple layers of active arm elimination. We previously designed an algorithm with two layers: one is faster but cannot tolerate corruptions and the second one is slower but more robust. To be agnostic to corruption, we need to plan for all possible amounts of corruption simultaneously. To achieve this, we introduce $\log T$ layers. Each layer is slower but more robust than the previous one. We achieve that by selecting the ℓ-th layer with probability $2^{-\ell}$. By the argument in the last section, if the corruption level is at most C, then each layer with $\ell \geq \log C$ will observe $O(1)$ corruption in expectation and at most $O(\log(kT/\delta)$ corruption with high probability (probability at least $1-\delta$). As a result, enlarging the confidence intervals by a logarithmic amount suffices to make the corresponding active arm elimination instances behave almost as stochastic.

Global eliminations. We couple the $\log T$ instances through what we call global eliminations. If arm a is eliminated by the ℓ-th layer, then we eliminate a in all layers $\ell' \leq \ell$. This is important to prevent us from pulling arm a too often. If arm a is suboptimal and the adversary is C-corrupted, then arm a eventually becomes eliminated in the $\ell^\star = \lceil \log C \rceil$ layer after $\tilde{O}\big(1/\Delta(a)^2\big)$ plays in that layer. Since layer $\ell^\star$ is selected with probability $2^{-\ell^\star}$, it takes $\tilde{O}\big(C/\Delta(a)^2\big)$ iterations until arm is eliminated globally, in which case we have total regret at most $\tilde{O}(C/\Delta(a))$ from that arm.

Multi-layer active arm elimination race. We now combine these ideas in an algorithm for the general agnostic case. We call it a *race* since we view it as multiple layers racing to pick the optimal arm. The less robust layers are faster so they arrive first and we select (mostly) according to them until more robust but slower layers correct or confirm the current selection of the best arm.

The algorithm keeps $\log T$ different instances of active arm elimination (extending the two layers of fast-slow active arm elimination). The ℓ-th instance has as state the empirical means of each arm $\tilde{\mu}^\ell(a)$, the number $n^\ell(a)$ of times each arm a was pulled and the set $\mathcal{I}^\ell$ of inactive arms. The width of the confidence interval for arm a in the ℓ-th layer is defined similarly to before as:

$$\mathsf{wd}^\ell(a) = \sqrt{\frac{\log(4kT \cdot \log T/\delta)}{n^\ell(a)}} + \frac{\log(4kT \cdot \log T/\delta) + 3}{n^\ell(a)}.$$

In each round t we sample $\ell \in \{1, \ldots, \log T\}$ with probability $2^{-\ell}$ (with the remaining probability we select layer 1). When layer ℓ is selected, we make a move in the active arm elimination instance corresponding to that layer: we sample the active arm in that layer with the least number of pulls, i.e., arm $a \in [k] \setminus \mathcal{I}^\ell$ minimizing $n^\ell(a)$. In case $[k] \setminus \mathcal{I}^\ell$ is empty, we pull an arbitrary arm from $[k] \setminus \mathcal{I}^{\ell'}$ for the lowest ℓ' such that $[k] \setminus \mathcal{I}^{\ell'}$ is non-empty.

To couple different layers, we ensure the following invariant. Once arm a' is eliminated in layer ℓ because there is another active arm a in layer ℓ such that $\tilde{\mu}^\ell(a) - \tilde{\mu}^\ell(a') < \mathsf{wd}^\ell(a) + \mathsf{wd}^\ell(a')$ we eliminate arm a' in all previous layers, keeping the invariant that: $\mathcal{I}^1 \supseteq \mathcal{I}^2 \supseteq \mathcal{I}^3 \supseteq \ldots$. Figure 3.1 provides an example of the state of the algorithm, which is formally defined in Algorithm 4. We again simplify notation there, by using a^t to denote the selected arm and omitting the dependence on t from empirical mean $\tilde{\mu}^\ell(a,t)$ and number of trials $n^\ell(a,t)$.

Subsequently we provide the main result, a regret guarantee for multi-layer active arm elimination race (Therorem 3.2).

Theorem 3.2. With probability $1 - \delta$, the multi-layer active arm elimination race

	arm 1	arm 2	$\cdots$	arm k
$\ell = 1$	🟥 $\tilde{\mu}^1(1), n^1(1)$	🟥 $\tilde{\mu}^1(2), n^1(2)$	🟥 $\cdots$	🟥 $\tilde{\mu}^1(k), n^1(k)$
$\ell = 2$	$\tilde{\mu}^2(1), n^2(1)$	🟥 $\tilde{\mu}^2(2), n^2(2)$	$\cdots$	🟥 $\tilde{\mu}^2(k), n^2(k)$
$\ell = 3$	$\tilde{\mu}^3(1), n^3(1)$	🟥 $\tilde{\mu}^3(2), n^3(2)$	$\cdots$	🟥 $\tilde{\mu}^3(k), n^3(k)$
$\vdots$	$\vdots$	$\vdots$		🟥 $\vdots$
$\ell = \log T$	$\tilde{\mu}^{\log T}(1), n^{\log T}(1)$	$\tilde{\mu}^{\log T}(2), n^{\log T}(2)$	$\cdots$	🟥 $\tilde{\mu}^{\log T}(k), n^{\log T}(k)$

Figure 3.1: Example of the state of the algorithm: for each layer ℓ and arm a we keep the estimated mean $\tilde{\mu}^\ell(a)$ and the number of pulls $n^\ell(a)$. Red cells indicate arms that have been eliminated in that layer. If an arm is eliminated in a layer, it is eliminated in all previous layers. If a layer where all the arms are eliminated (like layer 1 in the figure) is selected, we play an arbitrary active arm with the lowest layer that contains active arms.

has regret in the agnostic C-corrupted case bounded by:

$$\mathcal{O}\left(\sum_{a \neq a^*} \frac{k \cdot C \cdot \log(kT/\delta) + \log(T)}{\Delta(a)} \cdot \log(kT/\delta)\right).$$

Proof sketch. Similarly to the previous theorem, the final regret guarantee comes via summing over layers that are essentially stochastic and layers that are not robust to corruption. All the layers ℓ' with $C \leq 2^{\ell'}$ fall into the first category; similar to Lemma 3.3, with high probability the corruption they experience when the selected layer $\ell = \ell'$ is at most $c^\ell = \log(kT/\delta)$ as this corruption is subsampled. Each such layer incurs therefore regret of $\mathcal{O}\left(\frac{\log(kT/\delta)}{\Delta(a)}\right)$. Since there are at most $\log(T)$ such layers, the second term in the theorem is derived.

The challenge is to bound the regret incurred by layers that are not robust to the corruption (i.e. $2^\ell < C$). However, there exists some layer $\ell^\star = \min \ell : C \leq 2^\ell$ that is above the corruption level. By bounding the amount of steps that this level

Algorithm 4: Multi-layer Active Arm Elimination Race

```
Require: Number of arms $k$, horizon $T$
1: Initialize $n^\ell(a) = 0$, $\tilde{\mu}^\ell(a) = 0$, $\mathcal{I}^\ell = \emptyset$ for all $a \in [K]$ and $\ell \in \{1, \dots, \log T\}$.
2: For Rounds $t = 1..T$
3:     Sample algorithm $\ell \in \{1, \dots, \log T\}$ with probability $2^{-\ell}$. Else $\ell = 1$.
4:     If $[K] \setminus \mathcal{I}^\ell \neq \emptyset$
5:         Play arm $a^t \leftarrow \arg\min_{a \in [K] \setminus \mathcal{I}^\ell} n^\ell(a)$
6:         Update $\tilde{\mu}^\ell(a^t) \leftarrow [n^\ell(a^t) \cdot \tilde{\mu}^\ell(a^t) + \tilde{r}^t_{a^t}]/[n^\ell(a^t) + 1]$ and $n^\ell(a^t) \leftarrow n^\ell(a^t) + 1$
7:         While exists arms $a, a' \in [K] \setminus \mathcal{I}^\ell$ with $\tilde{\mu}^\ell(a) - \tilde{\mu}^\ell(a') > \mathsf{wd}^\ell(a) + \mathsf{wd}^\ell(a')$
8:             Eliminate $a'$ by adding it to $\mathcal{I}^{\ell'}$ for all $\ell' \leq \ell$
9:     Else
10:        Find minimum $\ell'$ such that $[K] \setminus \mathcal{I}^{\ell'} \neq \emptyset$; play arbitrary arm in that set
       without updating any estimate.
```

will require in order to eliminate each suboptimal arm $a \neq a^\star$ in the incorrect layers (via Lemma 3.2), we again obtain a bound on the regret caused by this arm in those layers. Since we take the minimum such layer and the tolerance of layers is within powers of 2, the fact that its corruption level does not match exactly the corruption that occurred only costs an extra factor of 2 in the regret. The details of the proof are provided in Appendix B.3. ■

3.6 Remarks

More information about the paper. The results presented in this chapter are joint work with Vahab Mirrokni and Renato Paes Leme [92]. So far, we discussed the results with respect to the uncorrupted objective (where we earn the uncorrupted rewards while observing the corrupted feedback). As we said, this captures settings where there exists some fraudulent activity that does not count towards the utility gained by the system, e.g. social welfare objective in online advertising with click fraud or recommender systems with fake reviews. Our results extend to the case where we also earn the corrupted rewards since the two objectives are within C of each other and we anyway have a linear degradation on C. This captures settings such as the revenue objective in click fraud (the platform gets the revenue even if it comes from undetected corrupted activity) or cases where the corruption is not malevolent but actually affects the experience in a non-i.i.d. manner (e.g. a construction next door affects a restaurant experience without having to do with the inherent quality of the restaurant).

Although for the uncorrupted objective, the linear degradation on C is unimprovable as we discussed in Section 3.2, for the corrupted objective the situation is less straightforward. For the notion of pseudoregret, we provide an improved dependence for some regimes of C (in fact, follow-up work can obtain an optimal dependence of $\sqrt{C}$ through the technique of Wei and Luo [127]; personal communication with the authors). On the other hand, for high-probability guarantees we show a lower bound showing that any algorithm that is optimal in the stochastic case (as the fast-slow active arm elimination) needs to degrade linearly with corruption even for the simpler setting where the input is either stochastic or C-corrupted case with a known level C. Understanding, whether an improved dependence on C can also be achieved with high probability at the expense of some logarithmic degradation in the stochastic case is an exciting open direction.

Related work regarding corruptions. Our work is one of the first trying to understand the effect of corruptions in stochastic bandit learning.

Prior to our work, there have been two such attempts. One direction is the best of both worlds [34, 120, 14, 119] which aims to provide a single algorithm achieving simultaneously the stochastic guarantee when the input is stochastic and the worst-case guarantee when it is adversarial. However, most of these works provide no handling for scenaria that are in between, which is the more common case. In fact, one of this works [120] has extended the guarantees to the mildly contaminated case where the input can be corrupted but the corruption cannot ever significantly alter the performance of different arms; in particular, the empirical gap between $a^\star$ and any other a can decrease by at most a factor of 2. This assumption does not hold in the main motivating applications such as click fraud where the adversary may completely corrupt the best arm in the initial rounds. Another approach aimed to add some stochastic corruptions to the feedback received to make bandit learning differentially private [56]; this is orthogonal to our main motivation where the corruptions are adversarial.

From the follow-up work, the most directly related is the one of Gupta et al. [61] who improve the guarantee presented in this chapter by decomposing the dependence on kC from the dependence on the gaps. This is done via a randomized scheme which works in phases and uses only samples from the previous phase at any time to limit the effect that any particular sample can have to the future. This plays a similar role to our multi-layer construction.

CHAPTER 4

ONLINE ALGORITHMS WITH PREDICTIONS

In scenaria where the algorithmic decisions alter the state of the system, the online learning framework is not directly applicable. For example, in bipartite matching, when an element is matched to a user, it becomes unavailable for future users. As a result, the notion of *best fixed action in hindsight* (of the regret benchmark) is ill-defined as there one cannot match this item multiple times.

To deal with this state-dependent decision-making, a better approach is to compare with the ex-post optimal solution (i.e. a benchmark that has the benefit of hindsight and makes decisions with full knowledge of the future). This is the classical benchmark that is mostly used in competitive analysis approaches, which is the prominent approach in the theoretical computer science literature. Note that it is a much stronger benchmark than the regret benchmark as it compares to the true ex-post optimal solution. Not surprisingly, it tends to utilize more structure in the underlying settings and does not allow the costs to change arbitrarily as in adversarial online learning discussed in Chapter 2.

One nice property of competitive analysis techniques is that they target worst-case instances without imposing distributional assumptions on the input. However, this property is often a pitfall of the method as the resulting algorithms tend to be overly cautious and, as a result, often cannot escape from the worst-case guarantees even when the input is nicely behaved. In the previous two chapters, we showed how to adapt online learning techniques to obtain improved performance when a the input has a nice structure in a way that is robust to the structure not holding. It is natural therefore to wonder whether something similar can be achieved for competitive analysis techniques.

In this chapter, we provide a way to enhance the performance of competitive analysis techniques when we have a particular well-behavedness in our input. In particular, we assume that we have access to a machine learned predictor (potentially erroneous). We assume that this predictor, when accurate, offers enough information to achieve offline optimality, i.e. the performance of the benchmark. Augmented with this predictor, we show how we can use this predictive power to enhance the performance when the predictor is relatively accurate, without sacrificing the worst-case robustness of the employed method.

4.1 Preliminaries on caching and competitive analysis

The caching problem. The caching (or online paging) problem considers a system with two levels of memory: a slow memory of size m and a fast memory of size k, which we refer to as cache. A caching algorithm is faced with a sequence of requests for elements. If the requested element is in the fast memory, a *cache hit* occurs and the algorithm can satisfy the request at no cost. If the requested item is not in the fast memory, a *cache miss* occurs, the algorithm fetches the item from the slow memory, and places it in the fast memory before satisfying the request. If the fast memory is full, then one of the items must be evicted. The eviction strategy forms the core of the problem. The goal is to find an eviction policy that results in the fewest number of cache misses.

Competitive analysis To obtain worst-case guarantees for an online algorithm (that must make decisions as each element arrives), we compare its performance to that of an offline optimum (that has the benefit of hindsight). Let σ be the input sequence of elements for a particular online decision making problem, $cost_A(\sigma)$ be the cost incurred by an online algorithm $\mathcal{A}$ on this input, and $\textsc{Opt}(\sigma)$ be the cost incurred by the optimal offline algorithm. Then algorithm $\mathcal{A}$ has competitive ratio $\textsc{cr}$ if for all sequences σ, $cost_{\mathcal{A}}(\sigma) \leq \textsc{cr} \cdot \textsc{Opt}(\sigma)$.

For the caching problem, the optimal offline algorithm at time t evicts the element from the cache that will arrive the furthest in the future; this is typically referred in the literature as Bélády's optimal replacement paging algorithm [22]. On the other hand, without the benefit of foresight, any deterministic caching algorithm achieves a competitive ratio of $\Omega(k)$, and any randomized caching algorithm achieves a competitive ratio of $\Omega(\log k)$ (see [100] for a nice exposition).

4.2 Caching augmented with a machine learned predictor

Online with Machine Learned Advice. Before focusing on how to enhance caching with a machine learned predictor, we specify a general framework to incorporate predictors in online algorithms. We term this framework *Online with Machine Learned Advice* or *OMLA* for a shortcut.

We first specify the input and the predictions made by the machine learned predictor $h \in \mathcal{H}$ from a class $\mathcal{H}$. The online input consists of a set of elements $\mathcal{Z}$. For a specific input σ, its elements are denoted by $z(\sigma_1), z(\sigma_2), \ldots$ and its length by $|\sigma|$. Formalizing the machine learning task, we assume a feature space $\mathcal{X}$ and a label space $\mathcal{Y}$. The i-th element $z(\sigma_i)$ has features $x(\sigma_i) \in \mathcal{X}$ and a label $y(\sigma_i) \in \mathcal{Y}$ that represents ground truth. For any element i, the predictor returns a predicted label $h(x(\sigma_i))$. To ease notation we will also denote this by $h(\sigma_i)$. We instantiate this framework to caching in the end of this section. To ease presentation, we assume that the mapping from features to labels is deterministic; our results extend to randomized mappings by applications of Jensen's inequality.

To measure the performance of the predictor h, we first define a loss function $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^{\geq 0}$. When the labels lie in a metric space, some examples of loss functions include absolute loss $\ell_1(y, \hat{y}) = |y - \hat{y}|$, squared loss $\ell_2(y, \hat{y}) = (y - \hat{y})^2$, and classification loss $\ell_c(y, \hat{y}) = \mathbf{1}_{y \neq \hat{y}}$. In defining the framework, we are not concerned with the semantics of the labels, i.e. what is the quantity that h is predicting or how it was trained – we are only interested in its performance. The error of the predictor h on a sequence σ with respect to loss function ℓ is therefore:

$$\eta_\ell(h, \sigma) = \sum_i \ell(y(\sigma_i), h(\sigma_i)).$$

Instantiated with the absolute loss function for the caching problem, the error of the predictor is $\eta_{\ell_1}(h, \sigma) = \sum_i |y(\sigma_i) - h(\sigma_i)|$. We use $\eta_1(h, \sigma)$ as shorthand for this.

Definition 4.1. *The* Online with Machine Learned Advice *(OMLA) model includes:*

- *An input $\sigma = \{z(\sigma_1), z(\sigma_2), \ldots, z(\sigma_{|\sigma|})$; each $z(\sigma_i) \in \mathcal{Z}$ has features $x(\sigma_i) \in \mathcal{X}$ and labels $y(\sigma_i) \in \mathcal{Y}$.*
- *A predictor $h : \mathcal{X} \to \mathcal{Y}$ that predicts a label $h(\sigma_i)$ for each $x(\sigma_i) \in \mathcal{X}$.*
- *The error of predictor h at sequence σ w.r.t. loss ℓ, i.e., $\eta_\ell(h, \sigma)$.*

Our goal is to create online algorithms that, when augmented with a predictor h, can use its advice to achieve an improved competitive ratio. To evaluate how well an algorithm $\mathcal{A}$ performs with respect to this task, we extend the definition of competitive ratio to be a function of the predictor's error. We first define the set of predictors that are sufficiently accurate.

Definition 4.2. *For a fixed optimization problem Π, let $\textsc{Opt}_\Pi(\sigma)$ denote the value of the optimal solution on the input σ. We say that a predictor h is ϵ-*accurate *with respect to a loss function ℓ for problem Π if for any σ,*

$$\eta_\ell(h, \sigma) \leq \epsilon \cdot \textsc{Opt}_\Pi(\sigma).$$

We will use $\mathcal{H}_\ell(\epsilon)$ to denote the class of ϵ-accurate predictors, omitting the quantifier on Π for notational clarity.

At first glance, it may appear unnatural to tie the error of the prediction to the value of the optimal solution. However, our goal is to have a definition that is invariant to simple padding arguments. For instance, consider a sequence $\sigma' = \sigma\sigma$, which concatenates two copies of an instance σ, and assume that the predictor makes the same (relative) predictions within each instance σ. It is clear that the prediction error of any predictor doubles, but this is not due to the predictor suddenly being worse. One could instead normalize the prediction error by the length of the sequence, but in many problems, including caching, one can artificially increase the length of the sequence without impacting the value of the optimum solution, or the impact of predictions. Instead, normalizing by the value of the optimum captures both of these problems.

Call an algorithm $\mathcal{A}$ *ϵ-assisted* if it has access to an ϵ-accurate predictor. The competitive ratio of an ϵ-assisted algorithm is itself a function of ϵ.

Definition 4.3. *Let $\mathrm{CR}_{\mathcal{A}(h)}(\sigma)$ be the competitive ratio of algorithm $\mathcal{A}$, which uses a ϵ-accurate predictor h with respect to a loss function ℓ, when applied on sequence σ. The* competitive ratio *of an ϵ-assisted algorithm $\mathcal{A}$ is:*

$$\mathrm{CR}_{\mathcal{A},\ell}(\epsilon) = \max_{\sigma, h \in \mathcal{H}_\ell(\epsilon)} \mathrm{CR}_{\mathcal{A}(h)}(\sigma).$$

We are now ready to define the desiderata that we wish our algorithm to satisfy. In particular, we would like our algorithm to perform as well as the offline optimum when the predictor is perfect, degrade gracefully with the error of the predictor, and perform as well as the best online algorithm regardless of the error of the predictor. More formally:

Definition 4.4. *$\mathcal{A}_h$ is β-*consistent *if* $\text{CR}_{\mathcal{A},\ell}(0) = \beta$.

Definition 4.5. *$\mathcal{A}$ is α-*robust *for a function $\alpha(\cdot)$, if* $\text{CR}_{\mathcal{A},\ell}(\epsilon) = O(\alpha(\epsilon))$.

Definition 4.6. *$\mathcal{A}$ is γ-*competitive *if* $\text{CR}_{\mathcal{A},\ell}(\epsilon) \leq \gamma$ *for all values of ϵ.*

Our goal is to find algorithms that simultaneously optimize aforementioned three properties. They are ideally 1-consistent, recovering the optimal solution when the predictor is perfect. They are $\alpha(\cdot)$-robust for a slow growing function α, ideally in the smaller possible rate – these algorithms can seamlessly handle errors in the predictor. Finally, they are also worst-case competitive and, even if the predictor completely fails, they perform as well as the best online algorithms without getting hurt by the predictor's inaccuracies.

Instantiating the framework to caching. To instantiate the framework for the caching problem, we need to define the oracle, the label space of the predictions, and their semantics. The element space $\mathcal{Z}$ corresponds to the universe of elements that may be requested. The input sequence $\sigma = \sigma_1, \sigma_2, \ldots, \sigma_n$ determines the actual sequence of elements that are requested (fixed in advance and oblivious to the choices of the algorithm but unknown to it). Each element $z(\sigma_i) \in \mathcal{Z}$ has corresponding features $x(\sigma_i)$. These can encapsulate everything that is known about $z(\sigma_i)$ at the time i, for example, the times this element arrived in the past. The exact choice of $\mathcal{X}$ is orthogonal to our setting, though of course richer features typically lead to smaller errors.

One of the design choices when adding ML advice to the problem is the question of what to predict. For caching problems, a natural candidate is predicting the next time a particular element is going to appear. When such predictions are perfect, the online algorithm can recover the best offline optimum [22]. Formally, the label space $\mathcal{Y}$ is a set of positions in the sequence, $\mathcal{Y} = \mathbb{N}^+$. Given a sequence σ, $y(\sigma_i) = \min_{t>i}\{\tau : x(\sigma_t) = x(\sigma_i)\}$. If the element is never seen again, we set $y(\sigma_i) = n + 1$. Note that $y(\sigma_i)$ is completely determined by sequence σ. We use $h(\sigma_i)$ to denote the outcome of the prediction on an element with features $x(\sigma_i)$.

4.3 Blindly following the predictor is insufficient

Before describing our algorithm, we show that combining the predictions with ideas from competitive analysis is to a large extent essential; blindly evicting the element that is predicted the furthest in the future by the predictor or simple modifications of this idea can result to poor performance both with respect to robustness and competitiveness.

Evicting element predicted the furthest in the future. An immediate way to use the predictor is to treat its output as truth and optimize based on the whole predicted sequence. This corresponds to the Bélády rule that evicts the element predicted to appear the furthest in the future. We refer to this algorithm as algorithm $\mathcal{B}$ (as it follows the Bélády rule). Since this rule achieves offline optimality, this approach achieves the consistency desideratum, i.e. if the predictor is perfect, this algorithm is ex-post optimal. Unfortunately this approach does not have similarly nice performance with respect to the other two desiderata. With respect to robustness, the degradation with the average error of the predictor is far from the best possible, while a completely unreliable predictor leads to unbounded competitive ratios, far from the ones of the best online algorithm.

Theorem 4.1. The competitive ratio of ϵ-assisted algorithm $\mathcal{B}$ is $\text{CR}_{\mathcal{B},\ell_1}(\epsilon) = \Omega(\epsilon)$.

This means that when the error of the predictor is much worse than the offline optimum, the competitive ratio becomes unbounded. With respect to robustness, the rate of decay is far from optimal as we will see in Section 4.4.

Proof of Theorem 4.1. We show that for every ϵ, there exist a sequence σ and a predictor h such that the absolute error $\eta_1(h, \sigma) \leq \epsilon \cdot \text{OPT}$ while the competitive ratio of algorithm $\mathcal{B}$ is $(\epsilon - 1)/2$. For ease of presentation, assume that $\epsilon > 3$.

Consider a cache of size $k = 2$ and three elements a, b, c; the initial configuration of cache is a, c. The sequence consists of repetitions of the following sequence with $\epsilon - 1$ requests. The actual sequence will be $abcbcbc\dots$ (a appears once and then b and c appear alternately for $(\epsilon - 1)/2$ times).

In any repetition, the predictor accurately predicts the arrival time of all elements apart from two: i) when element a arrives, it predicts it to arrive again in the next step (instead of in the first step of the next repetition) and ii) when b arrives for the last time in one repetition, it predicts it to arrive again in the fourth position of the next repetition (instead of the second). As a result, the absolute error of the predictor is ϵ ($\epsilon - 2$ error in the a-misprediction and 2 error in the b-misprediction).

The optimal solution has two mistakes per repetition (one to bring a in the cache and one to directly evict it afterwards). Instead, the algorithm never evicts a as it is predicted to arrive much earlier than the other, and therefore has $\epsilon - 1$ cache misses. This means that the competitive ratio of this algorithm is $\Omega(\eta_1(h, \sigma)/\textsc{Opt}(\sigma))$ which completes the proof. ■

Evicting elements with proven wrong predictions. The problem in the above algorithm is that algorithm $\mathcal{B}$ keeps too much faith in predictions that have been already proven to be wrong (as the corresponding elements are predicted to arrive in the past). It is tempting to "fix" the issue by evicting any element whose predicted time has passed, and evict again the element predicted the furthest in the future if no such element exists. We call this algorithm $\mathcal{W}$ as it takes care of wrong predictions. Formally, let $h(j, t)$ denote the last prediction about z_j at or prior to time t. At time t algorithm $\mathcal{W}$ evicts an arbitrary item from the set $S_t = \{j : h(j, t) < t\}$ if $S_t \neq \emptyset$ and $\arg\max_{z_i \in \text{Cache(t)}} h(i, t)$ otherwise. We show that algorithm $\mathcal{W}$ has similarly bad performance guarantees.

Theorem 4.2. The competitive ratio of ϵ-assisted algorithm $\mathcal{W}$ is $\text{CR}_{\mathcal{W},\ell_1}(\epsilon) = \Omega(\epsilon)$.

Proof. Consider a cache of size $k = 3$ and four elements a, b, c, d; the initial configuration of cache is a, b, c and then d arrives. The actual sequence consists of repetitions of the following sequence with $(\epsilon/2) + 1$ requests (for ease of presentation, assume that $\epsilon > 6$). The actual sequence is $dabcabc\ldots$ and is denoted by σ.

In any repetition, the predictor h accurately predicts the arrival time of element d but always makes mistake in elements a, b, c by predicting them to arrive two time steps earlier. As a result, the absolute error of the predictor is ϵ (error of 2 for any of the appearances of a, b, c).

The optimal solution has two mistakes per repetition (one to bring element d and one to evict it afterwards). Instead the algorithm always evicts elements a, b, c because they are predicted earlier than their actual arrival and are therefore evicted as ″wrong″ predictions. This means that the competitive ratio of this algorithm is also $\Omega(\eta_1(h, \sigma)/\textsc{Opt}(\sigma))$ which completes the proof. ■

Beyond blindly trusting the predictor. The common problem in both examples is that there is an element that should be removed but the algorithm is tricked into keeping it in the cache. To deal with this in practice, popular heuristics such as LRU (Least Recently Used) and FIFO (First In First Out) avoid evicting recent elements when some elements have been dormant for a long time. This tends to utilize nice locality properties leading to strong empirical performance (especially for LRU and its variant, LRU-2). However, such heuristics impose a strict eviction policy which leads to weak performance guarantees. Moreover, incorporating the information provided by the predictor becomes complicated.

Competitive analysis has also built on the idea of evicting dormant elements via developing algorithms with stronger theoretical guarantees such as Marker. In Section 4.4, we show how to incorporate predictions in the Marker algorithm to enhance its performance when the predictions are good while retaining the worst-case guarantees. Interestingly, via our framework, we can provide improved guarantees for the aforementioned heuristics such as LRU, improving their worst-case guarantees while retaining their practical performance (see Section 4.5).

4.4 Main result: Predictive Marker algorithm

We now present our main technical contribution, a prediction-based adaptation of the Marker algorithm [52]. This ϵ-assisted algorithm gets a competitive ratio of $2 \cdot \min(O(\sqrt{\epsilon}, 2H_k)$ where $H_k = 1 + \frac{1}{2} + \cdots + \frac{1}{k}$ denotes the k-th Harmonic number.

Classic Marker algorithm. We begin by recalling the Marker algorithm and the analysis of its performance. The algorithm runs in phases. At the beginning of each phase, all elements are unmarked. When an element arrives and is already in the cache, the element is marked. If it is not in the cache, a *random unmarked* element is evicted, the newly arrived element is placed in the cache and is marked. Once all elements are marked and a new cache miss occurs, the phase ends and we unmark all of the elements.

For the purposes of analysis, an element is called *clean* in phase r if it appears during phase r, but does not appear during phase $r-1$. In contrast, elements that also appeared in the previous phase are called *stale*. The marker algorithm has competitive ratio of $2H_k - 1$ and the analysis is tight [1]. We use a slightly simpler analysis that achieves competitive ratio of $2H_k$ below. The crux of the upper bound lies in two claims. The first relates the performance of the optimal offline algorithm to the total number of clean elements Q across all phases.

Claim 4.1 ([52])**.** Let Q be the number of clean elements. Then the optimal algorithm suffers at least $Q/2$ cache misses.

The second comes from bounding the performance of the algorithm as a function of the number of clean elements.

Claim 4.2 ([52])**.** Let Q be the number of clean elements. Then the expected number of cache misses of the marker algorithm is $Q \cdot H_k$.

Predictive Marker. The algorithm of [52] is part of a larger family of *marking* algorithms, which never evict marked elements when there are unmarked elements present. Any algorithm in this family has a worst case competitive ratio of k. Therefore pairing predictions with a marking style algorithm would avoid the pathological examples we saw previously.

A natural approach is to use predictions for tie-breaking, specifically evicting the element whose predicted next appearance time is furthest in the future. When the predictor is perfect (and has zero error), the stale elements never result in cache misses, and therefore, by Claim 4.1, the algorithm has a competitive ratio of 2. On the other hand, by using the Marker algorithm and not blindly trusting the oracle, we can guarantee a worst-case competitive ratio of k.

We extend this direction to further reduce the worst-case competitive ratio to $O(H_k)$. To achieve this, we combine the prediction-based tie-breaking rule with the random tie-breaking rule. Suppose an element e is evicted during the phase. We construct a blame graph to understand the reason why e is evicted. There are two cases: either it was evicted when a clean element c arrived, then we add a directed edge from e to c. Alternatively, it was evicted because a stale element s arrived, but s was previously evicted. In this case, we add a directed edge from e to s. Note that the graph is always a set of chains (paths). The total length of the chains represents the total number of evictions incurred by the algorithm during the phase, whereas the number of distinct chains represents the number of clean elements. We call the lead element in every chain *representative* and denote it by

$\omega(r, c)$, where r is the index of the phase and c the index of the chain in the phase.

Our modification is simple – when a stale element arrives, it evicts a new element in a prediction-based manner if the chain containing it has length less than H_k. Otherwise it evicts a random unmarked element. (Looking ahead to the analysis, this switch to uniform evictions results in at most H_k additional elements added to any chain during the course of the phase. This guarantees that the competitive ratio is at most $O(H_k)$ in expectation; we make the argument formal in Theorem 4.3. The key to the analysis is the fact that the chains are disjoint, thus the interactions between evictions can be decomposed cleanly. We give a formal version of the algorithm in Algorithm 5. For simplicity, we drop dependence on σ from the notation.

Analysis. To analyze the performance of the proposed algorithm, we begin with a technical definition that captures how slowly a loss function ℓ can grow.

Definition 4.7. *Let $A_T = a_1, a_2, \ldots, a_T$, be a sequence of increasing integers of length T, that is $a_1 < a_2 < \ldots < a_T$, and $B_T = b_1, b_2, \ldots, b_T$ a sequence of non-increasing reals of length T, $b_1 \leq b_2 \leq \ldots \leq b_T$. For a fixed loss function ℓ, we define its* spread *$S_\ell : \mathbb{N}^+ \to \mathbb{R}^+$ as:*

$$S_\ell(m) = \min\{T : \min_{A_T, B_T} \ell(A_T, B_T) \geq m\}$$

The following Lemma instantiates the spread metric for loss metrics we consider and is proved in the Appendix C.

Lemma 4.1. For absolute loss, $\ell_1(A, B) = \sum_i |a_i - b_i|$, the spread of ℓ_1 is $S_{\ell_1}(m) \leq \sqrt{5m}$. For squared loss, $\ell_2(A, B) = \sum_i (a_i - b_i)^2$, the spread of ℓ_2 is $S_{\ell_2}(m) \leq \sqrt[3]{14m}$.

We now prove the main theorem of this chapter.

Algorithm 5: Predictive Marker

```
Require: Cache $C$ of size $k$ initially empty ($C \leftarrow \emptyset$).
1: Initialize phase $r \leftarrow 1$, unmark all elements ($\mathcal{M} \leftarrow \emptyset$), and set round $i \leftarrow 1$.
2: Initialize clean element counter $q_r \leftarrow 0$ and tracking set $\mathcal{S} \leftarrow \emptyset$.
3: Element $z_i$ arrives, receive prediction $h_i$, and save prediction $p(z_i) \leftarrow h_i$.
4: if $z_i$ results in cache hit or the cache is not full ($z_i \in C$ or $|C| < k$)
5:     Add to cache $C \leftarrow C \cup \{z_i\}$ without evicting any element and go to step 22
6: if the cache is full and all cache elements are marked ($|\mathcal{M}| = k$)
7:     Increase phase ($r \leftarrow r + 1$), initialize clean counter ($q_r \leftarrow 0$)
8:     Save current cache ($C \rightarrow \mathcal{S}$) as set of elements possibly stale in new phase
9:     Unmark elements ($\mathcal{M} \leftarrow \emptyset$).
10: if $z_i$ is a clean element ($z_i \notin \mathcal{S}$)
11:     Increase number of clean elements: $q_r \leftarrow q_r + 1$.
12:     Initialize size of new clean chain: $n(r, q_r) \leftarrow 1$.
13:     Evict unmarked element with highest prediction: $e = \arg\max_{z \in C - \mathcal{M}} p(z)$.
14: if $z_i$ is a stale element ($z_i \in \mathcal{S}$)
15:     It is the representative of some clean chain, let $c$ be this chain: $z_i = \omega(r, c)$.
16:     Increase length of the clean chain $n(r, c) \leftarrow n(r, c) + 1$.
17:     if $n(r, c) \leq H_k$
18:         Evict unmarked element with highest prediction: $e = \arg\max_{z \in C - \mathcal{M}} p(z)$.
19:     else Evict a random unmarked element $e \in C - \mathcal{M}$.
20:     Update cache by evicting $e$: $C \leftarrow C \cup \{z_i\} - \{e\}$.
21:     Set $e$ as representative for the chain: $\omega(r, c) \leftarrow e$.
22: Mark new element ($\mathcal{M} \leftarrow \mathcal{M} \cup \{z_i\}$), increase round ($i \leftarrow i + 1$), go to step 3.
```

Theorem 4.3. Let a loss function ℓ with spread bounded by S_ℓ. If S_ℓ is concave, the competitive ratio of ϵ-assisted Predictive Marker PM is bounded by:

$$\mathrm{CR}_{\mathrm{PM},\ell}(\epsilon) \leq 2 \cdot \min\left(1 + 2S_\ell(\epsilon), 2H_k\right).$$

To prove this theorem, we first introduce an analogue of Claim 4.2, which decomposes the total cost into that incurred by each of the chains individually.

To aid in our analysis, we consider the following marking algorithm, which we call SM (Special Marking). Initially we simply evict an arbitrary unmarked element. At some point, the adversary designates an arbitrary element not in the cache as *special*. For the rest of the phase, upon a cache miss, if the arriving element is special, the algorithm evicts a *random* unmarked element and designates the evicted element as special. If the arriving element is not special, the algorithm proceeds as before, evicting an arbitrary unmarked element.

Lemma 4.2. Using algorithm SM, in expectation at most H_k special elements cause cache misses per phase.

Proof. Consider the unmarked elements in the cache that never reappear during the phase. If one of these is designated special, it will not cause another cache miss before the end of the phase. We turn our analysis to elements that will re-appear during the phase.

Let A denote the subset of these elements that may become special; this set dynamically shrinks over time as elements appear and become marked. At the time the first special element causes a cache miss, these are the elements that are not already marked in the cache that will reappear during the phase, as well as those outside the cache that will appear before the end of the phase. Order the elements in A in decreasing order of their first arrival time. Observe that at the outset A contains at most $k-1$ elements.

For any $i \in \{1, \ldots, k-1\}$, we define $\mathcal{E}_i$ as the event that an element becomes special when it is the i-th element in the active ordering. Our goal is to show that:

$$Pr[\mathcal{E}_i] \leq \frac{1}{i+1}. \tag{4.1}$$

A key to the analysis is the fact that once event $\mathcal{E}_i$ occurs, only elements arriving even later (i.e. those with lower index in the active set) can become special. Therefore, given Equation (4.1), we can bound the expected number of misses caused by special elements as:

$$1 + \sum_{i=1}^{k-1} \frac{1}{i+1} = H_k,$$

where the first term is due to the first special element arriving, the the second term is due to the events $\mathcal{E}_1$ through $\mathcal{E}_{k-1}$.

Consider the last time an element becomes special while there are more than i elements in the active ordering; let ω be the special element. As we argued above, until this point $\mathcal{E}_i$ could not have occurred.

Now consider the time that ω re-appears. If there are exactly i elements in the active set, the probability that $\mathcal{E}_i$ occurs is bounded by $\frac{1}{i+1}$. We may have selected either one of the first i active elements, or an element in the cache that never appears before the end of the phase (at least one such element must exist, otherwise there are no cache misses during the phase). If there are fewer than i elements, the probability of $\mathcal{E}_i$ occurring is 0. ■

We now provide the lemma that lies in the heart of our robustness property.

Lemma 4.3. For any loss metric ℓ, any phase r, the expected length of any chain is at most $1 + \mathcal{S}_\ell(\eta_{r,c})$ where $\eta_{r,c}$ is the cumulative error of the predictor on the elements in the chain and $\mathcal{S}_\ell$ is the spread of the loss metric.

Proof. The clean element that initiates the clean chain evicts one of the unmarked elements upon arrival. Since it does so based on the Belady rule, it evicts the element s_1 that is predicted to reappear the latest in the future. If the predictor is perfect, this element will never appear in this phase. If, on the other hand, s_1 comes back (is a stale element) let s_2 be the element it evicts, which is predicted to arrive the furthest among the current unmarked elements.

Suppose there are m such evictions: $s_1, s_2, \ldots, s_m$. The elements were predicted to arrive in reverse order of their evictions. This is because elements s_j for $j > i$ were unmarked and in the cache when element s_i got evicted; therefore s_i was predicted to arrive later. However, the actual arrival order is the reverse. If $\eta_{r,c}$ is the total error of these elements, setting the actual arriving times as the sequence A_T and the predicted ones as the sequence B_T in the definition of spread (Definition 4.7), it means that $m \leq S_\ell(\eta_{r,c})$. ■

Combining the two above lemmas, we can obtain a bound on the expected length of any chain.

Lemma 4.4. For any loss metric ℓ, any phase r, the expected length of any chain is at most $\min(1 + 2\mathcal{S}_\ell(\eta_{r,c}), 2\log k)$ where $\eta_{r,c}$ is the cumulative error of the predictor on the elements in the chain and $\mathcal{S}_\ell$ is the spread of the loss metric.

Proof. The proof follows from combining the two above lemmas. By Lemma 4.2, if the chain switches to random evictions, it incurs another H_k cache misses in expectation after the switch point (and its length increases by the same amount), capping the total length by $2H_k \leq 2\log k$. If the chain does not switch to random evictions, it has Belady evictions and, by Lemma 4.3, it incurs at most $\mathcal{S}_\ell(\eta_{r,c})$ misses from stale elements. ■

Proof of Theorem 4.3. Fix an arbitrary sequence of arrivals. Let Q be the number of clean elements (and therefore also chains). Any cache miss corresponds to a particular eviction in one clean chain; there are no cache misses not charged to a chain by construction. By Lemma 4.4, we can bound the evictions from clean chain c of the r-th phase by $\min(1 + 2 \cdot S_\ell(\eta_{r,c}), 2\log k)$. Since both S_ℓ and the minimum operator are concave functions, the way to maximize the length of chains is to apportion the total error, η, equally across all of the chains. Thus for a given error η and number Q of clean chains, the competitive ratio is maximized when the error in each chain is $\eta_{r,c} = \eta/Q$ each. The total number of stale elements is then: $Q \cdot \min(2 \cdot S_\ell(\eta/Q), 2H_k)$. By Claim 4.1, $\frac{Q}{2} \leq \textsc{Opt}(\sigma)$, implying the result since $\textsc{Opt} \leq Q$. ■

We now specialize the results for the absolute and squared losses.

Corollary 4.1. The competitive ratio of ϵ-assisted Predictive Marker with respect to the absolute loss metric ℓ_1 is bounded by $\mathrm{CR}_{\mathcal{PM},\ell_1}(\epsilon) \leq \min\Big(2 + 2 \cdot \sqrt{5\epsilon}, 4H_k\Big)$.

Corollary 4.2. The competitive ratio of ϵ-assisted Predictive Marker with respect to the absolute loss metric ℓ_2 is bounded by $\mathrm{CR}_{\mathcal{PM},\ell_1}(\epsilon) \leq \min\Big(2 + 2 \cdot \sqrt[3]{14\epsilon}, 4H_k\Big)$.

On the robustness rate of Predictive Marker. We show that our analysis is tight: any marking algorithm that uses the predictor in a deterministic way cannot achieve an improved guarantee with respect to robustness.

Theorem 4.4. Any deterministic ϵ-assisted marking algorithm $\mathcal{A}$, that only uses the predictor in tie-breaking among unmarked elements in a deterministic fashion, has a competitive ratio worse than $\mathrm{CR}_{\mathcal{A},\ell}(\epsilon) = \Omega(S_\ell(\epsilon))$ for all $\epsilon < k^2$.

Proof. Consider a cache of size k with $k+1$ elements. We will construct an instance that exhibits the above lower bound. Since $\mathcal{A}$ uses marking, we can decompose its analysis into phases. Let σ be the request sequence, and assume that we do not have any repetition of an item inside the phase; as a result the i-th element of phase r corresponds to element $\sigma_{(r-1)k+i}$.

Suppose the predictor is always accurate on elements 2 through $k - S_\ell(\epsilon) + 1$ in each phase. For the last $S_\ell(\epsilon) - 1$ elements of phase r as well as the first element of the of the next phase, the elements are predicted to come again at the beginning of the subsequent phase, at time $t = rk + 1$. Since the algorithm is deterministic, we order the elements so that their evictions are in reverse order of their arriving time. By the definition of spread, the error of the predictor in these elements is exactly ϵ and the algorithm incurs a cache miss for each. On the other hand, the offline optimum has only 1 miss per phase, which concludes the proof. ■

Theorem 4.4 establishes that the analysis of Predictive Marker is tight with respect to the rate of robustness, and suggests that algorithms that use the predictor in a deterministic manner may suffer from similar rates. However, a natural question that comes up is whether a better rate can be achieved using by a randomized marking algorithm better utilizing the predictor. Note that our algorithm uses randomization only once a competitive ratio of $\log k$ is already incurred. We conjecture that a rate of $\log(1 + \sqrt{\epsilon})$ with respect to the absolute loss is possible, similar to the exponential improvement randomized schemes obtain over the deterministic guarantees of k with respect to worst-case competitiveness.

4.5 Practical traits of the algorithm

In this section, we discuss some traits that make the algorithm more practical. In particular, we prove that our algorithm makes the errors of the predictor only have local negative effect and prevent them from propagating further. Subsequently, we show that common heuristic approaches, such as LRU, can be used as predictors in our framework. This allows us to combine their predictive power with robust guarantees when they fail.

Locality. The guarantee in Theorem 4.3 bounds the competitive ratio as a function of the quality of the prediction. One potential concern is that if the predictions have of a small number of very large errors, then the applicability of Predictive Marker may be quite limited.

Here we show that this is not the case. Due to the phase-based nature of the analysis, the algorithm essentially "resets" at the end of every phase, and therefore the errors incurred one phase do not carry over to the next. Moreover, the competitive ratio in every phase is bounded by $O(H_k)$.

Formally, for any sequence σ, we can define phases that consist of exactly k distinct elements. Let $\textsc{cl}(r, \sigma)$ be the number of clean elements in phase r of sequence σ, and let $\eta_{\ell,r}(h, \sigma)$ denote the error of predictor h restricted only to elements occurring in phase r.

Theorem 4.5. Consider a loss function ℓ with spread S_ℓ. If S_ℓ is concave, the competitive ratio of Predictive Marker PM at sequence σ when assisted by a predictor h is at most:

$$\mathrm{CR}_{\mathcal{PM},\ell} \leq \frac{\sum_r \mathrm{CL}(r,\sigma) \cdot \min(1 + 2S_\ell(\eta_{\ell,r}(h,\sigma), 2H_k)}{\sum_r \mathrm{CL}(r,\sigma)}$$

Proof. The proof follows directly from Lemma 4.4 and applying Jensen's inequality only within the chains of the phase (instead of also across phases as we did in Theorem 4.3). ■

This theorem illustrates a very nice property of our algorithm. If the predictor h is really bad for a particular chunk of time (has big locality in its errors) then the clean chains of the corresponding badly predicted phases will contribute the second term (the logarithmic worst-case guarantee) but the other phases will provide enhanced performance utilizing the predictor's advice. In this way, the algorithm adapts to the quality of the predictions, and bad errors do not propagate beyond the end of a phase. This quality is useful in caching where most patterns are generally well predicted but there are few unforeseen sequences.

Robustifying LRU. Another practical property of our algorithm is that it can seamlessly incorporate heuristics known to perform well in practice. In particular, the popular Least Recently Used (LRU) algorithm can be expressed within the Predictive Marker framework. Consider the following predictor h: when an element σ_i arrives at time i, h predicts next arrival time $h(\sigma_i) = -i$.[1]

Note that, by doing so, at any point of time, among the elements that are in the cache, the element predicted the furthest in the future is exactly the one that

[1]If we prefer positive numbers in the predictions, we can select $h(\sigma_i) = T - i$ where T is the number of requests.

has appeared the least recently. Also note that any marked element needs to have arrived later than any unmarked element. As a result, if we never switched to random evictions (or had k in the RHS of line 17 in Algorithm 5), the Predictive Marker algorithm assisted with the LRU predictor is exactly the LRU algorithm.

The nice thing that comes from this observation is that we can robustify the analysis of LRU. LRU, and its variants like LRU(2), tend to have very good empirical performance as using the recency of requests is a good predictor about how future requests will arise. However, the worst-case guarantee of LRU is unfortunately of the order of k since it is after all a deterministic algorithm. By expressing LRU as a predictor in the Predictive Marker framework and using a switching point of H_k for each clean chain, we exploit most of this predictive power while also guaranteeing a logarithmic worst-case bound on it.

4.6 Remarks

More information about the paper. The results presented in this chapter are joint work with Sergei Vassilvitskii [95]. In the paper, we also discuss further how to trade off worst-case competitiveness for enhanced robustness by adapting the switching threshold. Moreover, we show a general construction that can combine an algorithm that is robust with one that is worst-case competitive via multiple restarts of the algorithm. Although this construction is not very practical, it suggests that the biggest bottleneck in designing algorithms with our aforementioned desiderata lies in deriving algorithms that are robust, i.e. gracefully degrade with the error of the predictor. Finally, we show that our algorithm has good empirical performance, outperforming both LRU and Marker even without any modification in very simple datasets.

More generally on online algorithms with predictions. Our work has initiated a line of work that studies the design of online algorithms that are aided with a machine learned predictor and want to obtain improved guarantees when the prediction is accurate while being robust to imperfections. Follow-up works focus on dealing with such imperfections in predictions in online settings such as ski rental or job scheduling by Purohit et al. [107] and Mitzenmacher [99].

Beyond online algorithms, there have been a few works nicely injecting predictions in decision-making tasks. Medina and Vassilvitskii [97] show how to use such predictions in revenue optimization. Kraska et al. [84] demonstrate empirically that introducing machine learned components to classical algorithms (in their case index lookups) can result in significant speed and storage gains. Finally, Rakhlin and Sridharan [110] show how to enhance online learning guarantees when the losses can be predicted in a relatively accurate way.

CHAPTER 5

DYNAMIC PRICING IN RIDESHARING

Another place where state becomes important is ridesharing. When Uber or Lyft match a particular driver to a customer, the driver moves with the customer to her desired destination. This changes the underlying state of the system and may create an undesired spatial mismatch between supply (drivers) and demand (customers). To deal with this mismatch, the platforms have at their disposal some control levers such as pricing that can help them modulate the process.

Classical competitive analysis paradigms can apply in the above scenario but often disregard crucial complexities that are first-order effects in these systems. One such paradigm is the k-server problem (it extends the caching problem which we discussed in the previous chapter). There, requests arrive in different locations and the platform needs to send servers to deal with them aiming to minimize the aggregate delay. This captures the spatial component of ridesharing systems but optimizes a largely irrelevant objective: platforms typically do not aim to serve all users under high stress. Instead they care about objectives such as throughput (how many customers got served), social welfare (how much value their service created to society), or revenue (how much money the platform gained). Another classical competitive analysis paradigm is bipartite matching which also captures an important effect: matching a customer to a driver makes the latter unavailable to future requests. However, this paradigm ignores future network effects as the driver will eventually become again available to serve demand possibly in a different location. Finally, most competitive analysis results target worst-case arrival sequences, while in a ridesharing system there is much well-behaved stochasticity.

In this chapter, we depart from competitive analysis and develop a general queueing-theoretic framework capturing the stochasticity in user requests. Focusing on the steady-state performance of the system, we obtain approximation guarantees in such ridesharing systems via appropriately pricing different rides. Our queueing-theoretic approach for the setting is motivated by classical works on controls in state-dependent stochastic processes. This approach allows us to model most of the first-order effects in these systems such as the ones we discussed in the previous paragraph. Interestingly, our guarantees are also parametric and become better as the ratio of drivers to locations increases; they achieve asymptotic optimality and provide effective approximation guarantees for the real parameters of the systems. In particular, consider a ridesharing system with m drivers and n distinct locations (for instance, corresponding to Uber's Hexagonal Hierarchical Spatial Index). In this system, the approximation guarantee of our approach is $1 + (n-1)/m$, which is typically very close to 1 as there are significantly more drivers than locations in these systems.

5.1 Preliminaries on pricing without state

Myopic agents. The classical way to think about pricing is via assuming distributional knowledge about the values of the customers (demand elasticity). In the simplest setting, we wish to sell a digital good to a customer; this allows us to not care about future effects as digital goods have infinite supply. Let's assume that the customer has a value V drawn from a distribution $F(\cdot)$. Upon arrival, we can quote a price p and the customer is myopic, i.e. buys the good if her value is above the price ($V \geq p$) or leaves the system otherwise.

In the next section, we will extend this scenario in the settings where this decision may also impose future network state externalities to other users. In particular, we will assume that riders behave as myopic agents. They have a value to go from a source location to a destination. When quoted a price lower than their value, they pay it and perform the ride. The network externality component comes from the fact that this can only happen if there exist an available vehicles in the source location; this complicates the setting as we see in Section 5.2.

Revenue management without state. Coming back to the simpler setting, a natural question is: *What is the price that optimizes our expected revenue?* To answer the above question, we need to consider the two opposite forces that arise in this process. On the one hand, putting a higher price p increases the likelihood that we end up disswaying the customer from making the purchase – the purchase only happens with probability $1 - F(p)$. On the other hand, condtioned on him buying, a higher price means more revenue as we cash the price p. As a result, to optimize revenue, we need to find the price p that optimizes $p \cdot (1 - F(p))$.

Pricing for other objectives. Similarly, if we wanted to optimize throughput, we would need to select the price that optimizes $1 \cdot (1 - F(p))$ and clearly this corresponds to the smallest possible price so that all users want to buy the good at this price. Last, if we wanted to optimize social welfare, we would want to select the price that would optimize the expected value of the customer $\mathbb{E}_{V\sim F}[V 1_{V \geq p}] = \mathbb{E}[V | V \geq p] \cdot (1 - F(p))$. Again, in this case this is trivially optimized in the smallest possible price as it allows more users to buy. More generally, all these reward functions have the form: $\widetilde{R}(p) = \widetilde{I}(p) \cdot (1 - F(p))$, where $\widetilde{I}(p)$ corresponds to the expected reward from a customer conditioned on selling and is instantiated as following for different objectives:

- *Throughput*: $\widetilde{I}^T(p) = 1$.
- *Social welfare*: $\widetilde{I}^W(p) = \mathbb{E}_{V\sim F}[V|V \geq p]$.
- *Revenue*: $\widetilde{I}^R(p) = p$.

Quantiles and concave reward assumption. It is easier to express this quantity with respect to the *inverse demand* or *quantile* $q = 1-F(p)$. This denotes the fraction of customers who accept price p. For ease of presentation we assume that the density of F is positive everywhere in its domain (for which we only assume that it is contiguous and intersects with $(0, \infty)$), implying that there is a 1-1 mapping between prices and quantiles. As F is therefore invertible, we can write $p = F^{-1}(1-q)$. This allows us to abuse notation throughout this chapter by using prices and quantiles interchangeably. Further, we define $F(\infty) = 1$, that is, we assume we can set a price high enough so that an arbitrarily small (or even 0) fraction of customers is willing to pay it.

The reward function that the platform optimizes can be rewritten as a function of quantiles as: $R(q) = q \cdot I(q)$ where $I(q)$ is the quantile-equivalent of $\widetilde{I}(p)$ and corresponds again to the expected reward from a customer conditioned on selling. For the previous objectives, it is instantiated as:.

- *Throughput*: $I^T(q) = 1$.
- *Social welfare*: $I^W(q) = \mathbb{E}_{V\sim F_{ij}}[V|F(V) \geq 1-q]$.
- *Revenue*: $I^R(q) = F^{-1}(1-q)$.

We refer to $R(q)$ as the *reward curve* of the function for its reward objective of interest (analogous to the notion of revenue curves; cf. [66]). Our framework applies generally to any objective that satisfies the condition that $R(q)$ are concave in q, implying that $I(q)$ are non-increasing in q. Note that this always holds for throughput and welfare; for revenue, distributions fulfilling the condition are referred to as regular distributions (cf. Appendix D.1)).

5.2 Pricing as a control in queueing-theoretic networks

In ridesharing, pricing decisions are more complex as they affect each other through *network supply externalities*. In particular, a customer wishing to move between locations i to j does not only contribute to the objective directly but may also help future customers at j by moving the driver there (if the ride occurs).

Model. We therefore consider a system with m units (drivers or vehicles) and n nodes (locations or stations). Customers traveling between nodes i and j arrive at node i according to a Poisson process of rate ϕ_{ij}. As in the previous section, each customer traveling from i to j has a value drawn independently from a distribution $F_{ij}(\cdot)$. We assume that F_{ij} has a density and that each customer has a positive value with some probability, i.e. $F_{ij}(0) < 1$; all the other assumptions made in the previous section continue to hold for the value distributions. Upon arrival at i, a customer is quoted a price p_{ij}, and engages a unit to travel to j if her value exceeds this price, i.e. with probability $1 - F_{ij}(p_{ij})$, and if at least one unit is available at node i; else she leaves the system immediately. If a ride occurs, the unit moves to j and the customer leaves the system thereafter.

A continuous-time Markov chain tracks the number of units across nodes. At time $t \geq 0$, the *state* of the Markov chain $\mathbf{X}(t) = (X_1(t), \ldots, X_n(t))$ contains the number of units $X_i(t)$ present at each node i. The state space of the system is denoted by $\mathcal{S}_{n,m} = \left\{(x_1, x_2, \ldots, x_n) \in \mathbb{N}_0^n | \sum_i x_i = m\right\}$. Throughout the chapter we use $\mathbf{X}(t), X_i(t)$ to indicate random variables, and $\mathbf{x}, x_i$ to denote specific elements of the state space. Note that the state-space is finite and $|\mathcal{S}_{n,m}| = \binom{m+n-1}{n-1} = \Omega(m^n)$.[1] Since our focus is on the long-run average performance, i.e. system performance under the steady state of the Markov chain, we henceforth suppress the dependence on t for ease of notation. For ease of presentation, we assume that rides between nodes occur without delay.[2] In the context of our model, this translates into an instantaneous state transition from $\mathbf{X}$ to $\mathbf{X} - e_i + e_j$ when a customer engages a unit to travel from i to j (where e_i denotes the ith canonical unit vector).

Pricing policies and objectives. We consider pricing policies that select point-to-point prices p_{ij} as a *function of the overall state* $\mathbf{X}$. Formally, given arrival rates and demand elasticities $\{\phi_{ij}, F_{ij}(\cdot)\}$, we want to design a pricing policy $\mathbf{p}(\cdot) = \{p_{ij}(\cdot)\}$, where each $p_{ij} : \mathcal{S}_{n,m} \to \mathbb{R} \cup \{\pm\infty\}$ maps the state to a price for a ride between i and j. Equivalently, we want to select quantiles $\mathbf{q}(\cdot) = \{q_{ij}\}$ where each $q_{ij} : \mathcal{S}_{n,m} \to [0, 1]$. For a fixed pricing policy $\mathbf{p}$ with corresponding quantiles $\mathbf{q}$, the *effective demand stream* from i to j (i.e. customers traveling from node i to j with value exceeding p_{ij}) thus follows a (state-dependent) Poisson process with rate $\phi_{ij} q_{ij}(\mathbf{X})$. This follows from the notion of probabilistic thinning of a Poisson process – the rate of customers wanting to travel from i to j is a Poisson process of rate ϕ_{ij}, and each customer is independently willing to pay p_{ij} with probability

[1]Every state corresponds to a placement of $n-1$ stripes in between m circles with the number of units at node k corresponding to the number of circles in between the $(k-1)$-th and k-th stripe).

[2]In the paper we drop this assumption as we discuss in Section 5.6.

$q_{ij} = 1 - F_{ij}(p_{ij})$. State-dependent prices also allow us to capture unavailability by defining $q_{ij}(\mathbf{x}) = 0$ if $x_i = 0$ (i.e. a customer with origin i is always turned away if there are no units at that station; recall we defined $F_{ij}(\infty) = 1$). Thus, a pricing policy $\mathbf{p}$, along with arrival rates and demand elasticities $\{\phi_{ij}, F_{ij}(\cdot)\}$, determines the transitions of the Markov chain. Note that this is a finite-state Markov chain, and furthermore, is irreducible under weak assumptions on the prices and the demand (cf. Appendix D.2); hence, it has a unique steady-state distribution $\pi(\cdot)$ with $\pi(\mathbf{x}) \geq 0 \,\forall\, \mathbf{x} \in \mathcal{S}_{n,m}$ and $\sum_{\mathbf{x}\in\mathcal{S}_{n,m}} \pi(\mathbf{x}) = 1$.

Our goal is to design a pricing policy $\mathbf{p}$ to maximize the steady-state performance under various objectives. In particular, we consider objective functions that decompose into per-ride reward functions $I_{ij} : \mathbb{R} \to \mathbb{R}$, which correspond to the reward obtained from a customer engaging a ride between stations i and j at price p_{ij}. This can be instantiated for canonical objectives such as throughput, social welfare and revenue as in the previous section. For a given objective, our aim is to select a pricing policy $\mathbf{p}$, equivalently quantiles $\mathbf{q}$, that maximizes the steady-state rate of reward accumulation, given by

$$\text{OBJ}_m(\mathbf{q}) = \sum_{\mathbf{x}\in\mathcal{S}_{n,m}} \pi(\mathbf{x})\cdot\Big(\sum_{i,j} \phi_{ij}\cdot q_{ij}(\mathbf{x})\cdot I_{ij}(q_{ij}(\mathbf{x}))\Big) = \sum_{\mathbf{x}\in\mathcal{S}_{n,m}} \pi(\mathbf{x})\cdot\Big(\sum_{i,j} \phi_{ij}\cdot R_{ij}(q_{ij}(\mathbf{x}))\Big). \quad (5.1)$$

Intuitively, Equation (5.1) captures that at any node i, customers destined for j arrive via a Poisson process with rate ϕ_{ij}, and find the system in state $\mathbf{x} \in \mathcal{S}_{n,m}$ with probability $\pi(\mathbf{x})$. They are then quoted a price $p_{ij}(\mathbf{x})$ (corresponding to quantile $q_{ij}(\mathbf{x})$), and engage a ride with probability $q_{ij}(\mathbf{x})$. The resulting ride then contributes in expectation $I_{ij}(q_{ij}(\mathbf{x}))$ to the objective function. Recall that unavailability of units is captured by our assumption that $q_{ij}(\mathbf{x}) = 0$ whenever $x_i = 0$. Notice that the component of Equation (5.1) that captures the flow of units traveling from i to j can be written as

$$f_{ij,m}(\mathbf{q}) = \sum_{\mathbf{x}\in\mathcal{S}_{n,m}} \pi(\mathbf{x}) \cdot \phi_{ij} \cdot q_{ij}(\mathbf{x}). \tag{5.2}$$

State-independent pricing and closed queueing models. The Markov chain described in Section 5.2 has the structure of a *closed queueing network* (cf. [121, 77]), a well-studied class of models in applied probability (closed refers to the fact that the number of units remains constant; in open networks, units may arrive and depart from the system). Our analysis crucially relies on some classical results from the queuing theory literature, which we review in this section. Our presentation here closely resembles that of Serfozo [121]. One particular class of pricing policies is that of *state-independent* policies, wherein we set point-to-point prices $\{p_{ij}\}$ which do not react to the state of the system. As a consequence, we have a constant rate of units departing from any node i (at any time t when $X_i(t) > 0$); this rate is independent of the network state. The resulting model is a special case of a closed queueing model proposed by Gordon and Newell [60].

Definition 5.1. *A* Gordon-Newell network *is a continuous-time Markov chain on states $\mathbf{x} \in \mathcal{S}_{n,m}$, in which for any state $\mathbf{x}$ and any $i, j \in [n]$, the chain transitions from $\mathbf{x}$ to $\mathbf{x} - e_i + e_j$ at a rate $P_{ij}\mu_i\mathbf{1}_{\{x_i(t)>0\}}$, where $\mu_i > 0$ is referred to as the* service rate *at node i, and $P \geq 0$ as the* routing probabilities *satisfying $\sum_j P_{ij} = 1$.*

In other words, if units are present at a node i in state $\mathbf{x}$, then departures from that node occur according to a Poisson distribution with rate $\mu_i > 0$; conditioning on a departure, the destination j is chosen according to state-independent routing probabilities P_{ij}.

The Markovian dynamics resulting from state-independent pricing policies fulfill the conditions of Gordon-Newell networks: fixing a price p_{ij} (with corresponding q_{ij}) results in a Poisson process with rate $\phi_{ij}q_{ij}$ of arriving customers *willing to pay price p_{ij}*. These customers engage a unit only if one is available, else leave the system. Thus, given quantiles $\mathbf{q}$, the time to a departure from node i is distributed exponentially with rate $\mu_i = \sum_j \phi_{ij}q_{ij}$ when $X_i > 0$ and with rate 0 otherwise. Further, conditioned on an arriving customer having value at least equal to the quoted price, the probability that the customer's destination is j, is $P_{ij} = \phi_{ij}q_{ij}/\sum_k \phi_{ik}q_{ik}$, independent of system state.

Unlike state-dependent policies that can be very complex, state-independent policies are easier to reason about. In particular, one advantage of considering state-independent policies (and drawing connections with Gordon-Newell networks) is that the resulting steady-state distribution $\left\{\pi_{\mathbf{p},m}(\mathbf{x})\right\}_{\mathbf{x}\in\mathcal{S}_{n,m}}$ can be expressed in product form, as established by the Gordon-Newell theorem.

Theorem 5.1 (Gordon-Newell Theorem [60])**.** Consider an m-unit n-node Gordon-Newell network with transition rates μ_i and routing probabilities P_{ij}. Let $\{w_i\}_{i\in[n]}$ denote the invariant distribution associated with the routing probability matrix $\left\{P_{ij}\right\}_{i,j\in[n]}$, i.e. $w_i = \sum_{j=1}^n P_{ij}w_j$, and define the *traffic intensity* at node i as $r_i = w_i/\mu_i$. Then the stationary distribution is given by:

$$\pi(\mathbf{x}) = \frac{1}{G_m}\prod_{j=1}^{n}\left(r_j\right)^{x_j}, \tag{5.3}$$

where the Gordon-Newell normalization constant is $G_m = \sum_{\mathbf{x}\in\mathcal{S}_{n,m}}\prod_{j=1}^n\left(r_j\right)^{x_j}$.

To obtain some intuition for the form of Equation (5.3), consider a system with just one unit. If μ_i is equal to 1 for all i, then w_i exactly captures the stationary distribution of a simple random walk with routing matrix P. Further, changing μ_i at a node i does not affect how frequently the unit visits i, but it affects how much time the unit spends at i before departing again. The distribution above can now be seen to be the occupancy distribution induced by m independent random walks. For a formal proof of how this stationary distribution arises in the case of the Gordon-Newell network, we refer the reader to Serfozo [121].

We now show how the Gordon-Newell theorem can simplify the objective in Equation (5.1). Recall that for an m-unit system with state-independent policy $\mathbf{p}$ (and corresponding quantiles $\mathbf{q}$), we obtain a Gordon-Newell network with service rate $\sum_j \phi_{ij} q_{ij}$ and routing probabilities $\phi_{ij} q_{ij} / \sum_k \phi_{ik} q_{ik}$ at node i. Let $\{\pi(\mathbf{x})\}_{\mathbf{x} \in \mathcal{S}_{n,m}}$ be the corresponding steady-state distribution. Since $\mathbf{q}$ is no longer a function of the system state, we can no longer set $q_i = 0$ when $X_i = 0$. Instead, we define $A_{i,m}(\mathbf{q}) = \sum_{\mathbf{x} \in \mathcal{S}_{n,m}} \pi(\mathbf{x}) \mathbf{1}_{\{x_i > 0\}}$ as the steady-state *availability* of units at node i (i.e. the probability in steady-state that at least one unit is present at node i). From Equation (5.2), the state-independent *steady-state rate of units* moving from i to j then simplifies to $f_{ij,m}(\mathbf{q}) = A_{i,m}(\mathbf{q}) \cdot \phi_{ij} q_{ij}$. Further, from Equation (5.3), one can derive (see e.g. Proposition 1.33 and Equation 1.31 in [121])

$$A_{i,m}(\mathbf{q}) = (G_{m-1}(\mathbf{q}) / G_m(\mathbf{q})) \cdot r_i(\mathbf{q}). \tag{5.4}$$

Notice that $r_i(\mathbf{q})$ denotes the traffic intensity at node i under $\mathbf{q}$. To see how this is derived, note that the probability of at least one unit at i is given by $\frac{1}{G_m(\mathbf{q})} \sum_{\mathbf{x} \in \mathcal{S}_{n,m} : x_i > 0} \prod_{j=1}^{n} \left(r_j \right)^{x_j}$. Now since each term in the summation has at least one factor $r_i(\mathbf{q})$, we can pull that out, whereby the remaining summation corresponds to configurations of $m-1$ units over n nodes, thus adding up to $G_{m-1}(\mathbf{q})$. Again, we refer the reader to Serfozo [121] for a detailed proof. Now, the objective in Equation (5.1) can be written as

$$\textsc{Obj}_m(\mathbf{q}) = \sum_i A_{i,m}(\mathbf{q}) \cdot \left(\sum_j \phi_{ij} q_{ij} \cdot I_{ij}(q_{ij}) \right) = \sum_i f_{ij,m}(\mathbf{q}) I_{ij}(q_{ij}). \tag{5.5}$$

To ease notation, we omit the explicit dependence on m when clear from context.

The infinite-unit limit: The stationary distribution described above (for state-independent pricing policies) holds for any finite m. Moreover, it can be shown (cf. Theorem 3.18 in [121]) that as $m \to \infty$, the stationary distribution of the m-unit system (as specified in Equation (5.3)) converges in distribution to a limiting distribution. The exact form of this limiting system is well understood (cf. Section 3.7 in [121]), but not consequential for our work; one important fact to note about the limiting distribution is the existence of a node with availability 1. This captures the fact that in an infinite-unit system, at least one node must have an infinite number of units. While an analytical proof of this result can be found in Section 3.7 of [121], Lemma 5.6 can be interpreted as providing a combinatorial proof. Combined with Equation (5.4), this gives the following proposition.

Proposition 5.2. Recall that given $\mathbf{q} = \{q_{ij}\}$, the quantities $w_i(\mathbf{q})$ and $r_i(\mathbf{q})$ are independent of m. Then, given a policy with quantiles $\mathbf{q}$, in the infinite-unit limit, the steady-state availability of each node i is given by $r_i(\mathbf{q})/\max_j r_j(\mathbf{q})$.

Proof. To see this note that for any m, the maximum availability node $i^\star$ obeys $r_{i^\star}(\mathbf{q}) = \max_j r_j(\mathbf{q})$. Recall that $G_m(\mathbf{q})$ is the Gordon-Newell normalization constant of Theorem 5.1 and also recall the relation describing the availability of a node from Equation 5.4. As $m \to \infty$, the availability of the maximum availability node goes to 1, i.e. $(G_{m-1}(\mathbf{q})/G_m(\mathbf{q})) \cdot r_{i^\star}(\mathbf{q}) \to 1$. We thus obtain that $G_{m-1}(\mathbf{q})/G_m(\mathbf{q}) \to 1/r_i^\star(\mathbf{q})$, which in turn implies that, as $m \to \infty$,

$$A_{i,m}(\mathbf{q}) = (G_{m-1}(\mathbf{q})/G_m(\mathbf{q})) \cdot r_i(\mathbf{q}) \to r_i/r_{i^\star}(\mathbf{q}).$$

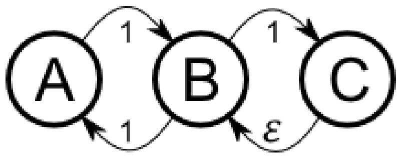
■

5.3 Network state externalities make pricing more complicated

Before describing our analysis in Section 5.4, we take a step back to consider the complications that these network externalities pose to the pricing task.

Ignoring state externalities can be problematic. As stressed before, the pricing decisions in the aforementioned setting have a double effect. On the one hand, they affect the reward we are getting from the current customer. On the other hand, they may alter the state of the system (configuration of units across nodes).

Ignoring this second effect can lead to very suboptimal decisions even in very simple examples. Consider a simple network with $n = 3$ nodes A, B, C and $m = 1$ unit. There is no demand between nodes A and C. The demand between most other pairs of nodes is pretty high ($\phi_{AB} = \phi_{BA} = \phi_{BC} = 1$). However, there is one pair of nodes where this does not happen: demand wishing to move from C to B is more anemic and is only $\phi_{CB} = \epsilon$ for some arbitrarily small $\epsilon > 0$. This is shown in the following figure (where the weights of the edges correspond to the demand that wants to move across the edges).



Suppose that we are interested in maximizing throughput. If we ignore the network effect of the decisions, we should set a very low price to ensure that any user that wants to move has value above the price. Hence, we would set $q_{AB} = q_{BA} = q_{BC} = q_{CB} = 1$. We now quantify the throughput of the system under this policy. The unit moves to C half the times it departs from B and, when this happens it remains there for a long time. Sending the unit to C is therefore a suboptimal decision as it would have been much better to set a large price in the direction $B \to C$ to ensure that the unit never goes to C and get stuck there. As a result, ignoring the network effects can lead to arbitrarily ineffective pricing schemes.

Non-concavity of objective function. To make things worse, the objective function that takes into account the network effect is not concave in quantiles (or prices) despite the fact that the reward curves are concave.

The intuition of the non-concavity can be seen in the same figure and comes from the following three observations. (i) The throughput is small (of order ϵ) when all quantiles are equal to 1. (ii) It is still almost as small when $q_{BC} = \frac{1+\epsilon}{2}$ and the other quantiles are 1. (iii) It is large when $q_{BC} = \epsilon$ and the other quantiles are 1. This contradicts concavity as the objective at $q_{BC} = \frac{1+\epsilon}{2}$ is smaller than the mean of the objectives at $q_{BC} = 1$ and $q_{BC} = \epsilon$ (keeping the other quantiles constant).

Explaining in more detail, when $q_{BC} = 1$, the single unit moves from B to C about half the times it departs from B; the expected time before an arrival at C is then equal to $(1/\epsilon)$, so it is not serving any rides for a long time. When $q_{BC} = \frac{1+\epsilon}{2}$ the unit still moves to C about one out of every three times it departs from B and then spends a lot of time waiting for an arrival at C. However, if $q_{BC} = \epsilon$, it only moves to C a fraction $\frac{\epsilon}{1+\epsilon}$ of the times it departs from B, canceling out the wait of length $1/\epsilon$ at C. In that case, the throughput is equal to $\frac{2+\epsilon}{3}$ (this also happens to be the solution our Algorithm 6 in Section 5.4 returns). Note that this also implies non-concavity in prices (e.g., if all passengers have uniform value distributions).

5.4 Main result: Elevated Flow Relaxation framework

In this section, we introduce our approximation framework which provides strong finite-unit performance guarantees. To convey the main ideas of our analysis, we first apply it to the most vanilla setting: using pricing as a control to maximize throughput. This is the only setting in which a finite guarantee exists in the literature (due to Waserhole and Jost [125]); we reprove this result with a different technique in order to illustrate the ingredients of our framework. In the next section, we show two more results showing the applicability of our framework. In fact, the framework is more widely applicable as we discuss in Section 5.6; we refer the interested reader to the paper [17] for other applications so that we do not confuse the presentation with many different models.

Even for the vanilla setting, there are two fundamental technical hurdles. First, we want to compare against state-dependent pricing policies in which the number of potentially distinct prices can be exponential in the number of units. Second, even if we restrict to state-independent pricing policies, the resulting problem is non-convex in the resulting quantiles as discussed in Section 5.3. We circumvent these hurdles by introducing a convex relaxation that restricts the policy search; in this setting the relaxation is the same as in [125]. The crux of our technical contribution is a novel approximation framework consisting of three steps. First, we bound the optimal objective by the objective of the convex relaxation. Next, we relate the convex relaxation to a system with infinitely many units. Finally, we compare the objective of a policy in the finite-unit system with its performance in the infinite-unit system.

Elevated Objective Function. We restrict our attention to state-independent pricing policies which reduces the output of the program from an exponential to a quadratic number of distinct prices (one for each source-destination pair). Recall from Equation (5.5) that the steady-state objective for state-independent policies can be written as

$$\textsc{Obj}_m(\mathbf{q}) = \sum_{i,j} f_{ij,m}(\mathbf{q}) \cdot I_{ij}(q_{ij})$$

where $f_{ij,m}(\mathbf{q}) = A_{i,m}(\mathbf{q}) \cdot \phi_{ij} q_{ij}$ are the resulting steady-state rates of units, which we also refer to as *steady-state flows*. For throughput, the objective is significantly simplified as $I^T_{ij}(q_{ij}) = 1$. Similarly, for a state-dependent policy, the objective is:

$$\textsc{Obj}_m(\mathbf{q}) = \sum_{\mathbf{x} \in S_{n,m}} \sum_{i,j} f_{ij,m}(\mathbf{q}(\mathbf{x})) \cdot I_{ij}(q_{ij}(\mathbf{x}))$$

Turning back our attention to state-independent policies, we first make a distinction between *actual quantiles* $q_{ij} = 1 - F_{ij}(p_{ij})$ and *effective quantiles* $\widehat{q_{ij}} = f_{ij,m}(\mathbf{q})/\phi_{ij} = A_{i,m}(\mathbf{q}) \cdot q_{ij}$. Whilst actual quantiles are in one-to-one correspondence to prices, effective quantiles incorporate thinning both due to the demand elasticity and due to the unavailability of units. Note that $\widehat{q_{ij}} \leq q_{ij}$; moreover, although any $q_{ij} \in [0, 1]$ can be induced via an appropriate price, not all $\widehat{q_{ij}}$ can be realized by prices as effective quantiles (e.g., if $m < n$, not all effective quantiles can be equal to 1). Since we assume that the per-ride rewards $I_{ij}(\cdot)$ are non-increasing on the quantile space, we have $I_{ij}(q_{ij}) \leq I_{ij}(\widehat{q_{ij}})$; for throughput this holds with equality, since $I^T_{ij}(\cdot) = 1$.

We now define the *elevated objective function* as

$$\widehat{\textsc{Obj}}(\mathbf{q}) = \sum_{i,j} \phi_{ij} q_{ij} I_{ij}(q_{ij}) = \sum_{i,j} \phi_{ij} R_{ij}(q_{ij}). \tag{5.6}$$

The elevated objective is essentially the objective of state-independent quantiles $\mathbf{q}$ ignoring demand thinning due to unavailability. It has two useful properties:

i) for all m and $\mathbf{q}$, the elevated objective upper bounds the true objective function, i.e. $\widehat{\text{OBJ}}(\mathbf{q}) \geq \text{OBJ}_m(\mathbf{q})$, and *ii*) it is a concave function of $\mathbf{q}$ (since we focus on objectives corresponding to concave reward curves $R_{ij}(\cdot)$).

The Flow Polytope. To make the elevated objective function relevant, we need to reinstate the effect of network externalities that we completely disregarded by ignoring unavailability. Therefore we impose a set of necessary (though not sufficient) properties that the steady-state rate of flows $\{f_{ij,m}(\mathbf{q})\}$ and the effective quantiles $\widehat{\mathbf{q}}$ must satisfy in order to be realizable. These properties form a linear polytope on these variables, which we refer to as *flow polytope* as it relates to flow conservation and capacity constraints. We begin by proving that both properties are indeed necessary.

Proposition 5.3 (Demand bounding)**.** For actual quantiles $\mathbf{q}$ under any state-dependent policy, the steady-state rate of flows fulfill the capacity bounding property $f_{ij,m}(\mathbf{q}) \leq \phi_{ij}$.

Proof. The proof follows immediately from $q_{ij}(\cdot) \leq 1$ which implies $\sum_{\mathbf{x}\in\mathcal{S}_{n,m}} \pi(\mathbf{x}) q_{ij}(\mathbf{x}) \leq 1$. ■

Proposition 5.4 (Supply Circulation)**.** For actual quantiles $\mathbf{q}$ under any state-independent policy, the steady-state rate of flows obey flow conservation $\sum_k f_{ki,m}(\mathbf{q}) = \sum_j f_{ij,m}(\mathbf{q})$ for every i, even for infinite-unit systems. Under state-dependent policies, flow conservation holds for any finite-unit system: $\sum_{\mathbf{x}\in S_{n,m}} \pi(\mathbf{x}) \cdot \sum_k f_{ki,m}(\mathbf{q}(\mathbf{x})) = \sum_{\mathbf{x}\in S_{n,m}} \pi(\mathbf{x}) \cdot \sum_j f_{ij,m}(\mathbf{q}(\mathbf{x}))$

Proof. We first consider state-independent policies. Recall that $w_i(\mathbf{q})$ is defined to be the leading left eigenvector of $\{P(\mathbf{q})\}_{i,j}$, where $P_{ij}(\mathbf{q}) = \phi_{ij} q_{ij} / \sum_j \phi_i j q_{ij}$. From this we get for all i:

$$\sum_j w_j(\mathbf{q}) \frac{\phi_{ji} q_{ji}}{\sum_k \phi_{jk} q_{jk}} = w_i(\mathbf{q}) = \frac{w_i(\mathbf{q})}{\sum_k \phi_{ik} q_{ik}} \left(\sum_k \phi_{ik} q_{ik} \right) \Rightarrow \sum_j r_j(\mathbf{q}) \phi_{ji} q_{ij} = \sum_k r_i(\mathbf{q}) \phi_{ik} q_{ik}$$

Multiplying both sides by $(G_{m-1}(\mathbf{q})/G_m(\mathbf{q}))$, and using our previous formula for the node availability (cf. Equation 5.4), we get $\sum_j A_{j,m}(\mathbf{q}) \phi_{ji} q_{ij} = \sum_k A_{i,m}(\mathbf{q}) \phi_{ik} q_{ik}$. Moreover, by Proposition 5.2 this also holds in the infinite-unit limit.

For state-dependent policies, we prove the claim via contradiction. Suppose that this does not hold. Then there exists a node that has incoming flow that is in steady-state less than the outgoing flow. As the system has finite number of units, this means that, after finite number of time, this node will have 0 units and then this can no longer be true which establishes the claim. ■

Interpreting the demand bounding and supply circulation properties in terms of *effective quantiles* of state-independent policies, we find the linear constraints

$$\widehat{q}_{ij} \in [0, 1] \text{ and } \sum_k \phi_{ki} \widehat{q}_{ki} = \sum_j \phi_{ij} \widehat{q}_{ij}.$$

Actual quantiles need not fulfill the supply circulation property, but any (state-independent) quantiles induce flows (effective quantiles) that fulfill it.

Pricing via the Elevated Flow Relaxation. Combining the elevated objective and the flow polytope, we obtain the *elevated flow relaxation program* (cf. Algorithm 6). For the case of maximizing throughput, this is a linear optimization problem since both objective function and polytope are linear. Recall from above that the supply circulation property implied flow conservation of the effective quantiles at each node. Algorithm 6 drops the availability term in the objective but imposes, as a constraint, the same demand bounding and flow conservation properties on the actual quantiles.

Algorithm 6: Elevated Flow Relaxation with Pricing for Throughput

```
Require: arrival rates $\phi_{ij}$, value distributions $F_{ij}$
1: Find $\left\{\widetilde{q}_{ij}\right\}$ that solve the following relaxation:
   $$\begin{array}{rcll} \max_{\mathbf{q}} & & \sum_{(i,j)} \phi_{ij} q_{ij} & \\ \sum_k \phi_{ki} q_{ki} & = & \sum_j \phi_{ij} q_{ij} & \forall\ i \\ q_{ij} & \in & [0,1] & \forall\ i,j. \end{array}$$
2: Output state-independent prices $\widetilde{p}_{ij} = F_{ij}^{-1}(1-\widetilde{q}_{ij})$ and respective quantiles $\widetilde{q}_{ij}$.
```

It is important to notice the distinction between the above constraint on actual quantiles and the supply circulation property of effective quantiles. The actual quantiles $\widetilde{q}_{ij}$ returned by the algorithm are *constrained* to satisfy $\sum_k \phi_{ki}\widetilde{q}_{ki} = \sum_j \phi_{ij}\widetilde{q}_{ij}$ at every node i. They then give rise to effective quantiles $\widehat{\mathbf{q}}$ (with $\widehat{q}_{ij} = A_{i,m}(\widetilde{\mathbf{q}}) \cdot \widetilde{q}_{ij}$) which obey the supply circulation property $\sum_k \phi_{ki}\widehat{q}_{ki} = \sum_j \phi_{ij}\widehat{q}_{ij}$ (as proven for any effective quantiles in Proposition 5.4). In other words, the linear program above restricts our pricing policy to induce actual quantiles which obey the flow circulation property, thereby mirroring a feature of effective quantiles. Henceforth, for any actual quantile $\mathbf{q}$, we refer to the property $\sum_k \phi_{ki} q_{ki} = \sum_j \phi_{ij} q_{ij}$ as the *demand circulation property*, in order to distinguish it from the supply circulation property for the effective quantiles.

The approximation guarantee. Our analysis relies on the three following lemmas. Lemma 5.1 shows that the solution of the elevated flow relaxation upper bounds the true objective of any state-depedendent pricing policy. Lemma 5.3 then shows that the true (non-elevated) objective of the pricing policy returned by our program is equal to the solution of the program, when applied to an infinite-unit system. Finally, Lemma 5.6 enables us to connect this solution to the true objective of the finite-unit system by showing that the true objective of any policy in the m-unit system is within a factor of $\frac{m}{m+n-1}$ of the objective in the infinite-unit system.

Lemma 5.1 (from finite-unit state-dependent to the elevated flow relaxation)**.** The value of the objective function of the optimal state-dependent policy is upper bounded by the value of the elevated objective function of the pricing policy $\widetilde{\mathbf{q}}$ returned by the elevated flow relaxation program:

$$\widehat{\text{OBJ}}(\widetilde{\mathbf{q}}) \geq \text{OPT}_m.$$

Proof. The optimal state-dependent pricing policy $\mathbf{q}^\star(\cdot)$ induces a steady-state distribution $\pi^\star(\cdot)$. Based on that distribution we define, analogously to the effective quantiles of state-independent policies, the average-fraction of customers that receive service for each origin-destination pair: $\mathbf{q} = \sum_{\mathbf{x}\in S_{n,m}} \pi^\star(\mathbf{x}) \cdot \mathbf{q}^\star(\mathbf{x})$. Then

$$\text{OPT}_m = \sum_{\mathbf{x}\in S_{n,m}} \pi^\star(\mathbf{x}) \sum_{i,j} \phi_{ij} q^\star_{ij}(\mathbf{x}) = \sum_{ij} \phi_{ij} q_{ij} = \widehat{\text{OBJ}}(\mathbf{q}).$$

We complete the proof by showing that $\widehat{\text{OBJ}}(\widetilde{\mathbf{q}}) \geq \widehat{\text{OBJ}}(\mathbf{q})$. To do so, we demonstrate that $\mathbf{q}$ is a feasible solution of the elevated flow relaxation program; this suffices as $\widetilde{\mathbf{q}}$ maximizes the elevated objective over the feasible set. We thus only need to show that $\mathbf{q}$ satisfies the demand bounding and demand circulation properties. The first holds trivially since $\mathbf{q}$ is a convex combination of $\mathbf{q}^\star(\cdot)$. The second holds true because $\mathbf{q}^\star$ induces steady-state flows that obey the supply circulation property (Proposition 5.4); hence, at every node i we have

$$\sum_k \phi_{ki} q_{ki} = \sum_k \sum_{\mathbf{x}\in\mathcal{S}_{n,m}} \pi^\star(\mathbf{x}) \cdot q^\star_{ki}(\mathbf{x}) \phi_{ki} = \sum_j \sum_{\mathbf{x}\in\mathcal{S}_{n,m}} \pi^\star(\mathbf{x}) \cdot q^\star_{ij}(\mathbf{x}) \phi_{ij} = \sum_j \phi_{ij} q_{ij},$$

which shows that $\mathbf{q}$ fulfills the demand circulation property. Hence $\mathbf{q}$ is a feasible solution to the elevated flow relaxation program and the result follows. ■

For the second step of our analysis, we use the following auxiliary lemma.

Lemma 5.2 (demand circulation property implies equal availabilities)**.** For any m (including ∞) if a state-independent pricing policy $\mathbf{q}$ satisfies the demand circulation property then, at all nodes i, the availabilities $A_{i,m}(\mathbf{q})$ are equal.

Proof. Consider $i^\star \in \arg\max A_{i,m}(\mathbf{q})$. Then the demand circulation and supply circulation properties imply

$$A_{i^\star,m}(\mathbf{q}) \sum_j \phi_{ji^\star} q_{ji^\star} = A_{i^\star,m}(\mathbf{q}) \sum_j \phi_{i^\star j} q_{i^\star j} = \sum_j A_{j,m}(\mathbf{q}) \phi_{ji^\star} q_{ji^\star}$$

and thus $\sum_j (A_{i^\star,m}(\mathbf{q}) - A_{j,m}(\mathbf{q}))\phi_{ji^\star} q_{ji^\star} = 0$. By choice of $i^\star$, each summand is nonnegative, so for each j such that $\phi_{ji^\star} q_{ji^\star} > 0$ we obtain $A_{j,m}(\mathbf{q}) = A_{i^\star,m}(\mathbf{q})$. All availabilities being equal then follows inductively using the assumption that our system is irreducible (cf. Appendix D.2). ■

Next we connect the elevated flow relaxation to the infinite-unit system. In fact, we show a stronger statement, that holds for any policy satisfying demand circulation and thus also for the solution of the elevated flow relaxation program.

Lemma 5.3 (from elevated flow relaxation to infinite-unit state-independent)**.** For any pricing policy $\mathbf{q}$ satisfying the demand circulation property, the value of the elevated objective function of $\mathbf{q}$ is equal to the value of its objective function in the infinite-unit system

$$\text{OBJ}_\infty(\mathbf{q}) = \widehat{\text{OBJ}}(\mathbf{q}).$$

Proof. Since $\mathbf{q}$ satisfies the demand circulation property, by Lemmas 5.2 and Proposition 5.2, the availability at all nodes is equal to 1. This means that (i) the value of the objective function in the infinite-unit limit for pricing policy $\mathbf{q}$ is equal to its elevated value (since no term was increased), and (ii) the flow of customers on each edge is equal to $\phi_{ij} \cdot q_{ij}$. ∎

For the third step of our framework, we introduce two auxiliary lemmas.

Lemma 5.4 (approximation of finite-unit equals maximum availability)**.** For any state-independent pricing policy $\mathbf{q}$, let $A_m(\mathbf{q}) = \max_i(A_{i,m}(\mathbf{q}))$ denote the maximum steady-state availability across all nodes. Then the objective function of $\mathbf{q}$ in the m-unit system is related to the infinite-limit objective as

$$\frac{\text{OBJ}_m(\mathbf{q})}{\text{OBJ}_\infty(\mathbf{q})} = r_{\max}(\mathbf{q}) \cdot \frac{G_{m-1}(\mathbf{q})}{G_m(\mathbf{q})} = A_m(\mathbf{q}).$$

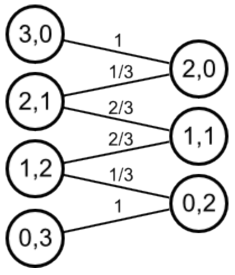
Proof. Let $B_i(\mathbf{q}) = \sum_j \phi_{ij} q_{ij} \cdot I_{ij}(q_{ij})$ denote the contribution of node i to the objective per unit of time in which station i is available. By substituting $A_{i,m}(\mathbf{q}) = (G_{m-1}(\mathbf{q})/G_m(\mathbf{q})) \cdot r_i(\mathbf{q})$, $A_{i,\infty}(\mathbf{q}) = r_i(\mathbf{q})/r_{\max}(\mathbf{q})$, and $B_i(\mathbf{q})$ into the definition of the objectives in Equation (5.5), we obtain

$$\frac{\text{OBJ}_m(\mathbf{q})}{\text{OBJ}_\infty(\mathbf{q})} = \frac{\sum_i A_{i,m}(\mathbf{q}) B_i(\mathbf{q})}{\sum_i A_{i,\infty}(\mathbf{q}) B_i(\mathbf{q})} = \frac{\frac{G_{m-1}(\mathbf{q})}{G_m(\mathbf{q})} \cdot \sum_i r_i(\mathbf{q}) B_i(\mathbf{q})}{\frac{1}{r_{\max}(\mathbf{q})} \cdot \sum_i r_i(\mathbf{q}) B_i(\mathbf{q})} = r_{\max}(\mathbf{q}) \cdot \frac{G_{m-1}(\mathbf{q})}{G_m(\mathbf{q})} = A_m(\mathbf{q}),$$

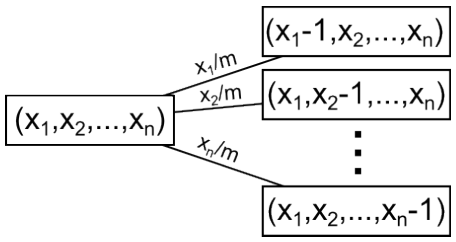
where the last equality follows from the characterization of the availabilities in Equation (5.4). Note that the argument relies on $OBJ_\infty(\mathbf{q}) \neq 0$ which holds for all policies/settings we consider. In particular, there is always a policy that charges $\epsilon > 0$ for every price and achieves a positive objective (since we assumed $F_{ij}(0) < 1$). ∎

Lemma 5.5 (weighted bipartite graph among state space of different-unit systems)**.** We call $\mathbf{y} \in \mathcal{S}_{n,m-1}$ a neighbor of $\mathbf{y} + e_i \in \mathcal{S}_{n,m} \forall i \in \{1, n\}$. There exists a weighted bipartite graph on $\mathcal{S}_{n,m} \cup \mathcal{S}_{n,m-1}$ such that i) an edge has non-zero weight only if it is connecting neighboring states , ii) for any vertex corresponding to a state in $\mathcal{S}_{n,m-1}$ the total weight of incident edges is equal to $\frac{m+n-1}{m}$, and iii) for any vertex corresponding to a state in $\mathcal{S}_{n,m}$ the total weight of incident edges is equal to 1.

Proof. Our construction is shown in the following figure. Each state $\mathbf{x} \in \mathcal{S}_{n,m}$ is adjacent to $\mathbf{y} = \mathbf{x} - e_i \in \mathcal{S}_{n,m-1}$ for all i with $x_i > 0$. On these edges, the weight is $\omega_{\mathbf{xy}} = \frac{x_i}{m}$. Thus, the total weight incident to $\mathbf{x}$ is $\sum_{\mathbf{y}} \omega_{\mathbf{xy}} = \sum_i \frac{x_i}{m} = 1$. On the other hand, each state $\mathbf{y} \in \mathcal{S}_{n,m-1}$ is adjacent to the states $\mathbf{x} = \mathbf{y} + e_i \; \forall i \in [n]$. The respective weight incident on $\mathbf{y}$ is $\sum_x \omega_{\mathbf{xy}} = \sum_i \frac{y_i+1}{m} = \frac{m-1+n}{m}$. Finally, there is positive weight on edges only between neighboring states. This concludes the proof of the lemma. ■



(a) Graph between $\mathcal{S}_{2,3}$ and $\mathcal{S}_{2,2}$



(b) Construction for general n, m

Lemma 5.6 (from infinite-unit to finite-unit state-independent)**.** For any state-independent pricing policy $\mathbf{q}$, the value of the objective of the policy $\mathbf{q}$ in the m-unit system is at least $m/(m+n-1)$ times the value of the objective of the same policy in the infinite-unit system.

$$\text{OBJ}_m(\mathbf{q}) \geq \frac{m}{m+n-1} \text{OBJ}_\infty(\mathbf{q}).$$

Proof. By Lemma 5.4, we have:

$$\frac{\textsc{Obj}_m(\mathbf{q})}{\textsc{Obj}_\infty(\mathbf{q})} = r_{\max}(\mathbf{q}) \cdot \frac{G_{m-1}(\mathbf{q})}{G_m(\mathbf{q})}.$$

In order to uniformly bound the above expression, we apply the weighted bipartite graph, between the states in $\mathcal{S}_{n,m-1}$ and the states in $\mathcal{S}_{n,m}$, described in Lemma 5.5. Following the same notation as before, we denote the weight between states $\mathbf{x} \in \mathcal{S}_{n,m}$ and $\mathbf{y} \in \mathcal{S}_{n,m-1}$ by $\omega_{\mathbf{xy}}$. Recall that non-zero weights only exist between neighboring states, i.e. when $\mathbf{x} = \mathbf{y} + e_i \in \mathcal{S}_{n,m}$ for some i; further, the total weight of edges incident to any state in $\mathbf{x} \in \mathcal{S}_{n,m}$ is equal to $\sum_{\mathbf{y}} \omega_{\mathbf{xy}} = 1$, and the total weight of edges incident to any state in $\mathbf{y} \in \mathcal{S}_{n,m-1}$ is equal to $\sum_{\mathbf{x}} \omega_{\mathbf{xy}} = \frac{m+n-1}{m}$.

$$\begin{aligned}
\frac{\textsc{Obj}_m(\mathbf{q})}{\textsc{Obj}_\infty(\mathbf{q})} = r_{\max}(\mathbf{p}) \cdot \frac{G_{m-1}(\mathbf{q})}{G_m(\mathbf{q})} &= r_{\max}(\mathbf{q}) \frac{\sum_{\mathbf{y}\in\mathcal{S}_{n,m-1}} \prod_{j=1}^n \big(r_j(\mathbf{q})\big)^{y_j}}{\sum_{\mathbf{x}\in\mathcal{S}_{n,m}} \prod_{j=1}^n \big(r_j(\mathbf{q})\big)^{x_j}} \\
&= r_{\max}(\mathbf{q}) \cdot \frac{\sum_{\mathbf{y}\in\mathcal{S}_{n,m-1}} \prod_{j=1}^n \big(r_j(\mathbf{q})\big)^{y_j}}{\sum_{\mathbf{x}\in\mathcal{S}_{n,m}} \Big(\sum_{\mathbf{y}\in\mathcal{S}_{n,m-1}} \omega_{\mathbf{xy}}\Big) \prod_{j=1}^n \big(r_j(\mathbf{q})\big)^{x_j}} \\
&= r_{\max}(\mathbf{q}) \cdot \frac{\sum_{\mathbf{y}\in\mathcal{S}_{n,m-1}} \prod_{j=1}^n \big(r_j(\mathbf{q})\big)^{y_j}}{\sum_{(\mathbf{x},\mathbf{y})\in\mathcal{S}_{n,m}\times\mathcal{S}_{n,m-1}} \omega_{\mathbf{xy}} \prod_{j=1}^n \big(r_j(\mathbf{q})\big)^{y_j+(x_j-y_j)}} \\
&\geq \frac{\sum_{\mathbf{y}\in\mathcal{S}_{n,m-1}} \prod_{j=1}^n \big(r_j(\mathbf{q})\big)^{y_j}}{\sum_{\mathbf{y}\in\mathcal{S}_{n,m-1}} \Big(\sum_{\mathbf{x}\in\mathcal{S}_{n,m}} \omega_{\mathbf{xy}}\Big) \prod_{j=1}^n \big(r_j(\mathbf{q})\big)^{y_j}} \\
&= \frac{\sum_{\mathbf{y}\in\mathcal{S}_{n,m-1}} \prod_{j=1}^n \big(r_j(\mathbf{q})\big)^{y_j}}{\Big(\frac{m+n-1}{m}\Big) \sum_{\mathbf{y}\in\mathcal{S}_{n,m-1}} \prod_{j=1}^n \big(r_j(\mathbf{q})\big)^{y_j}} = \frac{m}{m+n-1}
\end{aligned}$$

The third equality holds as $\sum_{\mathbf{y}} \omega_{\mathbf{xy}} = 1$; the second-to-last follows from $\sum_{\mathbf{x}} \omega_{\mathbf{xy}} = \frac{m+n-1}{m}$. Crucially, $\omega_{\mathbf{xy}} > 0$ only holds for neighboring states $\mathbf{x} \in \mathcal{S}_{n,m}$ and $\mathbf{y} \in \mathcal{S}_{n,m-1}$ implying the inequality. ■

Theorem 5.5 (approximation guarantee for pricing with throughput objective). Consider the throughput objective function $\textsc{Obj}_m^T$ for the m-unit system with concave reward curves $R_{ij}(\cdot)$. Let $\widetilde{\mathbf{q}}$ be the pricing policy returned by Algorithm 6 and $\textsc{Opt}_m$ be the value of the objective function for the optimal state-dependent pricing policy in the m-unit system. Then

$$\textsc{Obj}_m^T(\widetilde{\mathbf{q}}) \geq \frac{m}{m+n-1}\textsc{Opt}_m. \tag{5.7}$$

Proof. The proof follows by direct applications of Lemmas 5.1, 5.3, and 5.6. ■

5.5 Applying the framework to pricing beyond throughput

We now apply our framework to objectives beyond throughput. We first show how to maximize general objectives satisfying the concave reward curves assumption such as social welfare unconditonally and revenue with concave revenue curves (Appendix D.1). We then move our attention to multi-objective settings where we wish to maximize one objective function subject to a lower bound on another one, as is the case in Ramsey pricing [112].

General objectives with concave reward curves. We first provide the eleveated flow relaxation program for pricing with general objectives (Algorithm 7).

To extend our guarantee to objectives beyond throughput, we need a stronger version of Lemma 5.1 that does not rely on the linearity of the objective.

Lemma 5.7 (from finite-unit state-dependent to the elevated flow relaxation)**.** For objectives with concave price-setting reward curves $R_{ij}(\cdot)$, the value of the objective function of the optimal state-dependent policy is upper bounded by the value of the elevated objective function of the pricing policy $\widetilde{\mathbf{q}}$ returned by

```
Algorithm 7: Elevated Flow Relaxation with Pricing for General Objective
Require: arrival rates $\phi_{ij}$, value distributions $F_{ij}$, reward curves $R_{ij}$.
1: Find $\left\{\widetilde{q}_{ij}\right\}$ that solve the following relaxation:
       $\max_{\mathbf{q}} \quad \sum_{(i,j)} \phi_{ij} R_{ij}(q_{ij})$
       $\sum_k \phi_{ki} q_{ki} = \sum_j \phi_{ij} q_{ij} \quad \forall\, i$
       $q_{ij} \in [0,1] \quad \forall\, i,j.$
2: Output state-independent prices $\widetilde{p}_{ij} = F_{ij}^{-1}(1-\widetilde{q}_{ij})$ and respective quantiles $\widetilde{q}_{ij}$.
```

the elevated flow relaxation Program:

$$\widehat{\text{OBJ}}(\widetilde{\mathbf{q}}) \geq \text{OPT}_m.$$

Proof. Using the same notation as in the proof of Lemma 5.1, we have:

$$\text{OPT}_m = \sum_{\mathbf{x}\in S_{n,m}} \pi^\star(\mathbf{x}) \sum_{i,j} \phi_{ij} R_{ij}\Big(q_{ij}^\star(\mathbf{x})\Big) \leq \sum_{ij} \phi_{ij} R_{ij}\Big(\widehat{q}_{ij}\Big) = \widehat{\text{OBJ}}(\widehat{\mathbf{q}})$$

where the inequality holds by Jensen's inequality due to the concavity of the price-setting reward curves $R_{ij}(\cdot)$. The rest of the proof is identical to the proof of Lemma 5.1. ■

Theorem 5.6 (approximation guarantee for pricing with general objective)**.** Consider any objective function OBJ_m for the m-unit system with concave reward curves $R_{ij}(\cdot)$. Let $\widetilde{\mathbf{q}}$ be the pricing policy returned by Algorithm 7 and OPT_m be the value of the objective function for the optimal state-dependent pricing policy in the m-unit system. Then

$$\text{OBJ}_m(\widetilde{\mathbf{q}}) \geq \frac{m}{m+n-1}\text{OPT}_m. \tag{5.8}$$

Proof. The proof copies the one of Theorem 5.5 replacing Lemma 5.1 by Lemma 5.7. ■

Multi-objective settings. We now discuss how to derive bicriterion approximations in multi-objective optimization settings, in which one objective is maximized subject to a lower bound on another. Formally, the problem is as follows: we are given a m-unit system, a requirement $c \geq 0$, and objectives $\Phi_m(\cdot)$ and $\Psi_m(\cdot)$; the goal is to maximize $\Phi_m(\mathbf{q})$ subject to $\Psi_m(\mathbf{q}) \geq c$. We again assume that both objectives can be expressed by concave reward curves $\left\{R_{ij}^{\Psi}\right\}$ and $\left\{R_{ij}^{\Phi}\right\}$ respectively.

Similarly to Equation (5.6), we first elevate both objectives to obtain $\widehat{\Phi}(\widehat{\mathbf{q}}) = \sum_{i,j} \phi_{ij} R_{ij}^{\Phi}(\widehat{q}_{ij})$ and $\widehat{\Psi}(\widehat{\mathbf{q}}) = \sum_{i,j} \phi_{ij} R_{ij}^{\Psi}(\widehat{q}_{ij})$. Since per-ride rewards are non-increasing on the quantiles, this can only increase the values of the objectives. We then impose the supply circulation and demand bounding constraints to create the flow polytope constraints. This mathematical program (Algorithm 8) is the elevated flow relaxation program for our multi-objective setting; we argue below that this is indeed a relaxation. It can be efficiently optimized since the objective is concave while the polytope is convex: the convex combination of any two feasible quantiles is feasible since $\widehat{\Psi}(\cdot)$ is concave.

Theorem 5.7 (approximation guarantee for multi-objective pricing)**.** Let Φ_m and Ψ_m be objectives for the m-unit system with concave reward curves. Then the solution $\widetilde{\mathbf{q}}$ returned by Algorithm 8 is a (γ, γ) bicriterion approximation for the multi-objective pricing problem where $\gamma = m/(m+n-1)$, i.e. $\Phi_m(\widetilde{\mathbf{q}}) \geq \gamma \textsc{Opt}_m$ and $\Psi_m(\widetilde{\mathbf{q}}) \geq \gamma \cdot c$.

Proof. Let $\widehat{\mathbf{q}}$ denote the optimal solution of an auxiliary program where we only elevate the objective Φ (but not Ψ), i.e. we maximize $\widehat{\Phi}(\cdot)$ subject to $\Psi_m(\cdot) \geq c$ as

```
Algorithm 8: Elevated Flow Relaxation with Multi-objective Pricing
Require: arrival rates $\phi_{ij}$, value distributions $F_{ij}$, reward curves $R^{\Phi}_{ij}$ and $R^{\Psi}_{ij}$,
    requirement $c$.
1: Find $\{\widetilde{q}_{ij}\}$ that solve the following relaxation:
       $\max_{\mathbf{q}} \quad \widehat{\Phi}(\mathbf{q})$
       $\sum_k \phi_{ki} q_{ki} = \sum_j \phi_{ij} q_{ij} \quad \forall\, i$
       $q_{ij} \in [0,1] \quad \forall\, i, j.$
       $\widehat{\Psi}(\mathbf{q}) \geq c$
2: Output state-independent prices $\widetilde{p}_{ij} = F^{-1}_{ij}(1 - \widetilde{q}_{ij})$ and respective quantiles $\widetilde{q}_{ij}$.
```

well as the demand circulation and demand bounding constraints. Moreover, let $\mathbf{q}^\star$ denote the optimal state-dependent solution of the original (non-elevated) problemW. Then, for the first guarantee, we have:

$$\Phi_m(\widetilde{\mathbf{q}}) \geq \gamma\widehat{\Phi}(\widetilde{\mathbf{q}}) \geq \gamma\widehat{\Phi}(\widehat{\mathbf{q}}) \geq \gamma\Phi(\mathbf{q}^\star) = \gamma\textsc{Opt}_m$$

The first inequality is a simple application of Lemmas 5.3 and 5.6, since $\widetilde{\mathbf{q}}$ satisfies demand circulation and we can thus apply the lemmas to the objective $\Phi_m(\cdot)$. The second inequality holds since any solution of the auxiliary program is a feasible solution of the elevated flow relaxation. In particular, since the elevated objective $\widehat{\Phi}(\cdot)$ is pointwise no less than the original objective $\Phi_m(\cdot)$, the corresponding constraint in the auxiliary program is tighter. The last inequality holds by the same argument as in Lemma 5.7.

Regarding the second guarantee, we have:

$$\Psi_m(\widetilde{\mathbf{q}}) \geq \gamma\widehat{\Psi}(\widetilde{\mathbf{q}}) \geq \gamma c$$

The first inequality is again an application of Lemmas 5.3 and 5.6, while the second holds since $\widetilde{\mathbf{q}}$ is a feasible solution of the elevated flow relaxation and therefore satisfies its last constraint. ■

Note that the same approach yields multicriterion approximation algorithms for settings in which more than one constraint of the form $\Psi_m(\cdot) \geq c$ is given.

5.6 Remarks

More information about the paper. The results presented in this chapter are joint work with Sid Banerjee and Daniel Freund [17]. In this work, we also show that the approximation framework described in this chapter goes beyond the simple pricing setting described in this chapter and can extend to multiple different directions. For example, it can capture different rebalancing controls (beyond pricing) such as empty-vehicle rebalancing and matching customers to drivers. It can also extend to incorporate travel-times (instead of assuming that all travels are instantaneous). Finally, it can address constrained settings such as pricing that is only origin-based. All these results stem from the three-step approach described in Section 5.4.

On the approximation ratio. Our main result is an approximation framework that provides a state-independent policy with a $m/(m+n-1)$ approximation guarantee in steady-state. Our guarantee holds for a large class of objectives (revenue, throughput, welfare), controls (pricing, matching, empty-vehicle rebalancing), and constraints (multi-objective settings, prices coming from discrete price sets, travel-times). We note, that for the special case of maximizing throughput via pricing without constraints, Waserhole and Jost [125] provide the same guarantee although their analysis cannot extend more generally; our policy is the same as theirs for the special case. They also showed that, for this special case, the approximation ratio for a policy based on this relaxation is tight.

Our work shows the first universal performance bounds for a wide variety of controls and settings, and has inspired follow-up work that tries to improve the bounds in specific settings. Notable among these are two works. The first due to Qian et al. [108] shows how to obtain stronger guarantees, approximation ratio of $1 - e^{-\Theta(m)}$, for the matching control in settings obeying an additional *complete resource pooling* condition (a relaxed version of Hall's condition) via reverting to state-dependent policies. The second due to Balseiro et al. [16] demonstrates how to achieve better dependence on n, approximation ratio of $1 - o(1)$ competitive ratio when $n = \theta(m)$, for particular networks (star networks). However, understanding the limits and relative strenghts of different algorithms for particular controls and settings still remains an exciting open direction.

CHAPTER 6
EFFICIENCY OF DYNAMIC LEARNING OUTCOMES

In multi-agent systems, the platform is not the only entity that utilizes data in a way to enhance its decision-making. This data is also available to the other participants of the system who can learn from it and adapt their behavior. Consider online advertising as an example. Advertisers repeatedly bid for ad opportunity in various queries, observe their overall allocation and payment. This enables them to adapt their future bidding to better adjust to the competition.

To analyze agents' behavior in such complex systems, we ideally wish for a behavioral assumption that is easily achievable and weak enough so that it does not prescribe strict behavioral rules. One such behavioral assumption is that agents perform not much worse than what they would have had if they committed to a fixed strategy throughout all rounds. This is natural as, if agents repeatedly perform worse than the best fixed strategy, they can soon realize this fact and follow the better strategy. It is also easily satisfiable by multiple online learning algorithms such as the ones we described in Chapter 2; crucially though this assumption does not require agents to use any particular algorithm but rather just to satisfy the no-regret guarantee. Finally, empirical evidence supports this behavioral assumption in settings such as online advertising [102].

In this chapter, we aim to understand the efficiency of these dynamic learning outcomes, with respect to the total happiness of all participating entities, referred to as *social welfare*. Quantifying the inefficiency that selfish behavior causes to these systems, measured by the so called *Price of Anarchy*, is one of the cornerstones of algorithmic game theory. When the game is completely static and agents have converged to the so called Nash equilibrium of the one-shot game we now have a good understanding of this effect which, in many cases, is only a small deterioration [117, 37, 123]. Recently, these results have been extended to learning outcomes assuming that all agents employ online learning algorithms via extensions theorems based on a property referred to as *smoothness* [29, 116, 123]. However, these results still make an unrealistic strong assumption, requiring that the game played is completely identical across rounds. This is clearly not the case in applications like online advertising where there is significant turnover both in the player set and in their valuations of different outcomes. This chapter provides a general theory to make the aforementioned extension theorems robust even in settings where such population churn is present and very frequent.

6.1 Preliminaries on efficiency of selfish outcomes

Games and mechanisms. We consider game settings played repeatedly, where the population of players is evolving over time (as described in Section 6.2). Let G be an n-player normal form *stage game* and assume that game G is played repeatedly for T rounds.[1] Each player i has a strategy space S_i, with $\max_i |S_i| = N$, a type $v_i \in \mathcal{V}_i$ and a cost function $c_i(s; v_i)$ that depends on the strategy profile $s \in \times_i S_i$, and on her type. We denote with $C(s; v) = \sum_{i\in[n]} c_i(s; v_i)$ the social cost, where s is a strategy profile and v a type profile. Our main application described in this chapter concerns the case when the stage game is a utility maximization mechanism M, which takes as input a strategy profile and outputs an allocation $X_i(s)$ for each player and a payment $P_i(s)$. We assume that players have quasi-

[1]Although the game rules remain unaltered across rounds, the changes in participants affect the payoff matrix.

linear utility $u_i(s; v_i) = v_i(X_i(s)) - P_i(s)$ and the welfare is the sum of valuations (sum of utilities of bidders and revenue of auctioneer): $W(s; v) = \sum_{i\in[n]} v_i(X_i(s))$.

Disregarding the selfish behavior of the participants, we can define the optimal solution of the underlying optimization problem. Let $\mathbf{X}^n$ be the "feasible solution space" of the setting without incentives. In combinatorial auctions (the application in this chapter), this corresponds to the set of feasible partitions of items to bidders, while in network routing games it is the set of feasible integral flows. We overload the social cost and welfare notations and, for a feasible solution (or allocation) $x \in \mathbf{X}^n$, use $C(x; v)$ and $W(x; v)$ to denote the social cost or welfare of the solution[2]. We denote the optimal social cost or welfare for a type profile v as $\textsc{Opt}(v) = \min_{x\in\mathbf{X}^n} C(x; v)$ and $\textsc{Opt}(v) = \max_{x\in\mathbf{X}^n} W(x; v)$ respectively.

Application: Simultaneous first-price auctions. The application we discuss in this chapter concerns simultaneous first-price auctions. There are m items (e.g. ad opportunity) that the n players compete for. Players repeatedly participate in item auctions for each such item by submitting individual bids. The item gets assigned to its highest bidder and this player pays her bid (first-price) – ties are broken arbitrarily. Our results also extend to other auction formats such as second-price as well as hybrid auctions.

Each player cares about her individual utility which, as discussed before, is assumed to be quasilinear: equal to her valuation from acquiring items minus the price she pays. We use $v_i^t(A)$ to denote the valuation of the i-th player at time t if she obtains all items in set A; her valuation when not obtaining items is $v_i^t(\emptyset) = 0$. Valuations are additive across time but, at each round, users are unit-demand:

[2]Overloading the notation does not create ambiguity assuming the strategy sets S_i are disjoint from the possible solutions $\mathbf{X}$.

if they get more than one item in a single round, their valuation is equal to the maximum value they have among these items while they still pay the price for all. We denote by $v_i^t(j)$ the value of an item j for buyer i at time t; as a result, the player's value for a set A is $v_i^t(A) = \max_{j \in A} v_i^t(j)$. While we mostly focus on the unit-demand assumption, some of our results extend to more general valuation functions (see Section 6.5). We also assume that valuations are normalized to be in $[0, 1]$ and all non-zero valuations are at least $\rho > 0$ for some small constant ρ.

Regarding the players' bidding, we assume that the bids on each item comes from a sufficiently fine discrete bidding spice. We assume that the bids are always multiples of $\delta \cdot \rho$ for some small $\delta > 0$, where ρ is the minimum value from an item defined in the previous paragraph. Finally, we assume that players never bid for more than one item. As a result, the number of strategies available to each player is $N = \frac{m}{\delta \cdot \rho}$ (deciding whict item to target and how much to bid).

No-regret learning in games. In settings like the above it is clear that bidding the true valuation is not a good idea since this can only lead to utility 0: either the player wins the item and pays her valuation as a price, or she loses and gets again utility 0. Instead, we assume that the players employ strategies that provide them a good performance in the long run. In particular, we assume that they satisfy the no-regret learning properties described in Section 2.1. Recall the definition of ϵ-approximate regret from Section 2.2 (defined for a fixed $\epsilon > 0$):

$$ApxReg(f, \epsilon) = (1 - \epsilon) \sum_{t=1}^{T} \ell_{A(t)}^t - \sum_{t=1}^{T} \ell_f^t,$$

where $\ell_{A(t)}^t$ denotes the loss of of the learner with respect to the selected action $A(t)$ and ℓ_f^t denotes the loss with respect to a comparator f respectively.

Instantiated to a game setting, the ϵ-approximate regret of player i compared to a fixed strategy $s_i^\star$ will be shorthanded as $ApxReg_i\big(s_i^\star, \epsilon\big) = (1 - \epsilon)\sum_{t=1}^{T} c_i(s^t; v_i) - \sum_{t=1}^{T} c_i(s_i^\star, s_{-i}; v_i)$ for a cost minimization game. Respectively, for a utility-maximization mechanism, we define approximate regret as:

$$ApxReg_i(s_i^\star, \epsilon) = \sum_{t=1}^{T} u_i(s_i^\star, s_{-i}; v_i) - (1 + \epsilon)\sum_{t=1}^{T} u_i(s^t; v_i).$$

Solution-based Smoothness in Games and Mechanisms. Smooth games were introduced by Roughgarden [116] as a general framework bounding the price of anarchy in games. He also showed that smoothness based price of anarchy bounds extend to outcomes in repeated games where the set of players is fixed throughout the period and all players use no-regret learning.

We require a somewhat more general variant of smooth games, that compares the cost or utility resulting from a strategy choice to the social welfare of a specific solution, rather than comparing to the social optimum. For two strategy vectors s and $s^\star$ we use $(s_i^\star, s_{-i})$ to denote the vector where player i uses strategy $s_i^\star$ and all other players j use their strategy s_j.

Definition 6.1 (Solution-based smooth game)**.** *A cost-minimization game G is solution-based (λ, μ)-smooth for $\lambda > 0$ and $\mu < 1$, if for any feasible solution $x \in \mathbf{X}^n$, any type profile v and any player i, there exists a strategy $s_i^\star \in S_i$ depending only on her type v_i and her part of the solution x_i such that for any strategy profile s*

$$\sum_i c_i(s_i^\star(v_i, x_i), s_{-i}; v_i) \le \lambda C(x; v) + \mu C(s; v)$$

This means that, when players satisfy the no-regret property (as described in Section 2.1), the average social cost of these no-regret learning outcomes is bounded by $\frac{\lambda}{1-\mu}$OPT as time grows large.

In particular, we present this result for ϵ-approximate regret learners as this allows us to extend it in the next section. We assume that learners have expected ϵ-approximate regret of order $O(\log(NT)/\epsilon)$ which is the case for almost all adversarial online learning algorithms. For simplicity, we omit the O-notation.

Proposition 6.1. If a game is solution-based (λ, μ)-smooth expected ϵ-approximate regret of $\mathbb{E}\Big[ApxReg_i(s_i^\star, \epsilon)\Big] \leq \frac{\log(NT)}{\epsilon}$ for all $s_i^\star$, the average expected cost is at most $\frac{\lambda}{1-\mu-\epsilon} \cdot C(x; v) + \frac{1}{1-\mu-\epsilon} \cdot \frac{n\log(NT)}{\epsilon \cdot T}$ for all feasible solutions x.

Proof. Let s^t be the strategy vector at round t. Adding the ϵ-approximate regret inequalities for each player, and applying the smoothness property:

$$\begin{aligned}\frac{1}{T}\sum_t \mathbb{E}[C(s^t; v)] &= \frac{1}{T} \cdot \frac{1}{1-\epsilon}\sum_t \sum_{i\in[n]}(1-\epsilon) \cdot \mathbb{E}[c_i(s^t; v_i)] \\ &\leq \frac{1}{T} \cdot \frac{1}{1-\epsilon} \cdot \mathbb{E}\left[\sum_t \sum_i c_i(s_i^\star(v_i, x_i), s_{-i}^t; v_i) + \sum_i ApxReg_i(s_i^\star(v_i, x_i), \epsilon)\right] \\ &\leq \frac{1}{1-\epsilon} \cdot \left(\lambda C(x; v) + \mu \cdot \frac{1}{T}\sum_t \mathbb{E}[C(s^t; v)] + \frac{1}{T} \cdot \frac{n\log(NT)}{\epsilon}\right)\end{aligned}$$

The claimed bound follows by rearranging terms. ■

Note that it was crucial that strategy $s_i^\star$ was fixed across rounds since we used that player i does not regret not deviating to it. Since $s_i^\star$ is a function of the underlying optimization problem, this will no longer be the case in an evolving population as departures of players change the underlying optimization problem and may affect $s_i^\star$. We tackle this in the next section.

The application of this chapter is about mechanisms. For mechanisms we use the version of the smoothness definition of Syrgkanis and Tardos which assumes that all players have quasi-linear utilities. We again define a mechanism smooth *with respect to a solution* x, and allow the choice of strategy $s^\star$ to depend only on the player's part of the solution x_i and her type v_i. More formally:

Definition 6.2 (Solution-based smooth mechanism). *A mechanism $\mathcal{M}$ is solution-based (λ, μ)-smooth for $\lambda, \mu \geq 0$, if for any feasible solution $x \in \mathbf{X}^n$, any valuation profile v and any player i, there exists a deviating strategy $s_i^* \in S_i$ depending only on v_i and x_i such that for any strategy profile s,*

$$\sum_i u_i(s_i^*(v_i, x_i), s_{-i}; v_i) \geq \lambda W(x; v) - \mu \text{REV}(s).$$

where $\text{REV}(s) = \sum_{i=1}^n P_i(s)$.

Syrgkanis and Tardos [123] proved that a (λ, μ)-smooth mechanism has price of anarchy bounded by $\max(\mu, 1)/\lambda$, and the average social welfare of no-regret learning outcome is also at least $(\lambda/\max(\mu, 1))\text{OPT}(v)$. For example, simultaneous first-price auctions are $(\frac{1}{2}, 1)$-smooth implying efficiency of 2. Analogously:

Proposition 6.2. If a mechanism is solution-based (λ, μ)-smooth and players satisfy expected ϵ-approximate regret of $\mathbb{E}\left[ApxReg_i(s_i^\star, \epsilon)\right] \leq \frac{\log(NT)}{\epsilon}$ $\forall s_i^\star$, the average expected social welfare is at least $\frac{\lambda}{\max(\mu, 1+\epsilon)} \cdot W(x; v) - \frac{1}{\max(\mu, 1+\epsilon)} \cdot \frac{n \log(NT)}{\epsilon \cdot T}$ for all feasible solutions $x \in \mathbf{X}^n$.

Proof. Let s^t be the strategy vector at round t and $U(s^t; v^t)$ denote the welfare that comes from the players' utilities (and not the designer's revenue which is denoted by $\text{REV}(s^t)$). Adding the ϵ-approximate regret inequalities for each player, and applying the smoothness property:

$$\begin{aligned}
\frac{1}{T} \sum_t \mathbb{E}[U(s^t; v)] &= \frac{1}{T} \cdot \frac{1}{1+\epsilon} \sum_t \sum_{i \in [n]} (1+\epsilon) \cdot \mathbb{E}[u_i(s^t; v_i)] \\
&\geq \frac{1}{T} \cdot \frac{1}{1+\epsilon} \cdot \mathbb{E}\left[\sum_t \sum_i u_i(s_i^\star(v_i, x_i), s_{-i}^t; v_i) - \sum_i ApxReg_i(s_i^\star(v_i, x_i), \epsilon)\right] \\
&\geq \frac{1}{1+\epsilon} \cdot \left(\lambda W(x; v) - \mu \cdot \frac{1}{T} \sum_t \mathbb{E}[\text{REV}(s^t)] - \frac{1}{T} \cdot \frac{n \log(NT)}{\epsilon}\right)
\end{aligned}$$

The claimed bound comes from rearranging terms and noting that $W(s^t; v) = U(s^t; v^t) + \text{REV}(s^t)$ ■

6.2 Shifting learning and efficiency of dynamic outcomes

Dynamic population model. We focus on repeated game settings when the population evolves over time. Our model is formalized in the next definition.

Definition 6.3 (Repeated game/mechanism with dynamic population)**.** *A **repeated game with dynamic population** consists of a stage game G played for T rounds. Let P^t denote the set of players at round t, where each player $i \in P^t$ has a private type v_i^t. After each round, every player independently exits the game with a (small) probability $p > 0$ and is replaced by a new player with an arbitrary type. The utility of players is additive across rounds. This repeated game is denoted by $\Gamma = (G, T, p)$; similarly $\mathcal{M} = (M, T, p)$ denotes a corresponding mechanism.*

Our model of dynamic population assumes that after each round every player independently exits the game with a turnover probability $p > 0$; each player is expected to participate in $1/p$ rounds. To keep our model simple, we assume that when a player exits, she is replaced by a new participant. This assumption guarantees that there are exactly n players in each iteration, with a p fraction of the population changing each round in expectation. We make no assumption on the types of the new arriving players which can be selected adversarially.

To simplify notation, we use *player i* to denote the current i-th player, where this player is replaced by a new i-th player with probability p each round. An alternate view of the dynamic player population is to think of players as changing types after each iteration with a small probability p. We refer to such a change as *player i switches* or *turns over*.

Approximate regret with shifting comparators. To deal with a shifting environment, online learning guarantees against a comparator fixed throughout the whole time horizon are not strong enough to provide meaningful efficiency guarantees. Instead, we make a slightly stronger behavioral assumption, requiring that players have, for each time interval, low approximate regret against a comparator fixed within this interval. This allows us to have different comparators for different intervals. More formally, in utility settings, the adaptive approximate regret for interval $[\tau, \tau')$ is defined as:

$$AdApxReg(s_i^\star, \epsilon, \tau, \tau') = \sum_{t=\tau}^{\tau'-1} u_i(s_i^\star, s_{-i}^t; v_i) - (1+\epsilon) \sum_{t=\tau}^{\tau'} u_i(s^t; v_i^t).$$

When $\tau = 1$ and $\tau' = T$, this recovers the original approximate regret notion but it allows flexibility to have different comparators for different intervals; if the number of these comparators is not too large, the resulting guarantees are strong. This robustness against changing comparators dates back to Herbster and Warmuth [69]. The adaptive regret guarantee that compares to inteval-based fixed comparators was introduced by Hazan and Seshadhri [68] and further studied by Luo and Schapire [91] and Daniely et al. [43]. For the approximate regret version of the question, which will be useful in our setting, the bounds for each interval are of the form: $\mathbb{E}\Big[AdApxReg\Big(s_i^{\star,1:T}, \epsilon, \tau, \tau'\Big)\Big] = O\Big(\frac{\log(N\tau')}{\epsilon}\Big)$. This is satisfied by many full-feedback algorithms, e.g. sleeping experts algorithms such as the one of Blum and Mansour [30] for loss settings, or variants of multiplicative weights such as Noisy Hedge [9, 53] for both utilities and losses. We will again omit the O-notation from the bound to simplify notation and we will refer to the behavioral assumption that imposes that this is satisfied as: *the players satisfy the adaptive approximate regret property*.

Efficiency of dynamic learning outcomes. We now provide an efficiency result for learners that satisfy the adaptive approximate regret property. We also require the underlying game to satisfy one more property, which is satisfied for example in simultaneous first-price auctions when players do not bid over their value. This property establishes that players with no items in the feasible allocation will have literally no regret against a deviating strategy that attempts to "win" the empty allocation and not only a regret that vanishes over time on average.

Property 1. *The utility of any player i who is not allocated a resource is always nonnegative, i.e. $u_i(s; v_i) \geq 0$ for any strategy that is used by the players.*

Under this property, we provide the efficiency guarantee which we instantiate in the remainder of the chapter. Let $x^1, \dots, x^T$ denote a sequence of benchmark solutions. To denote the number of times that either the solution of player i changes, or her type changes when she is previously allocated a resource, we use

$$K_i(x_i^{1:T}) = 1 + \sum_{t=2}^{T} \mathbf{1}\Big[\Big(x_i^t \neq x_i^{t-1}\Big) \text{ or } \Big(x_i^{t-1} \neq \emptyset \text{ and } v_i^t \neq v_i^{t-1}\Big)\Big].$$

Unlike Proposition 6.2, we compare against a β-approximately optimal benchmark instead of the optimal one, for reasons that become clear in the next section. Note that the optimal solution is now changing with time as a function of player turnover. With no turnover and $\beta = 1$, Theorem 6.3 reverts to Proposition 6.2.

Theorem 6.3. Consider a repeated mechanism with dynamic population $\mathcal{M} = (M, T, p)$, such that the stage mechanism M is solution-based (λ, μ)-smooth, satisfies Property 1, and utilities are in $[0, 1]$. Suppose that there exists a randomized sequence $(v^{1:T}, x^{1:T})$ such that x^t is feasible (pointwise) and β-approximately optimal (in-expectation) for each t, i.e. $\beta \cdot \mathbb{E}[W(x^t; v^t)] \geq \mathbb{E}[\textsc{Opt}(v^t)]$. If players satisfy the adaptive approximate regret property:

$$\sum_t \mathbb{E}[W(s^t; v^t)] \geq \frac{\lambda}{\beta \cdot \max(\mu, 1+\epsilon)} \sum_t \mathbb{E}[\textsc{Opt}(v^t)] - \sum_i \frac{\mathbb{E}\Big[K_i\big(x_i^{1:T}\big)\Big] \cdot \log(NT)}{\max(\mu, 1+\epsilon) \cdot \epsilon}$$

where m is such that for any feasible allocation x, $|\{i : x_i \neq \emptyset\}| \leq m$.

Proof. In a dynamic population game, the underlying optimization problem is changing over time. Therefore the smoothness analysis described in the proof of Proposition 6.2 requires a stronger learning property. To deal with this, we define time-dependent deviating strategies that are related to the underlying optimization problem of the round. We then use the adaptive learning property to show that the players do not regret any sequence of such deviating strategies.

Let $s_i^{\star,t}$ be the deviation $s_i^\star(v_i^t, x_i^t)$ defined by the smoothness property and $s_i^{\star,1:T}$ be the sequence of these deviations. Since the deviating strategy $s_i^{\star,t}$ is determined by the allocation and type of the player, $K_i(x_i^{1:T})$ is an upper bound on the number of times that $s_i^{\star,1:T}$ changes. Let $r_i(s_i^{\star,1:T}, s^{1:T}; v^{1:T})$ be the approximate regret that player i has compared to selecting $s_i^{\star,t}$ at every round, i.e.:

$$r_i(s_i^{\star,1:T}, s^{1:T}; v^{1:T}) = \sum_{t=1}^{T} u_i(s_i^{\star,t}, s_{-i}^t; v^t) - (1+\epsilon) \sum_{t=1}^{T} u_i(s^t; v^t) \tag{6.1}$$

For shorthand, we denote this with $r_i^\star$ in this proof.

Let $\tau_{i,r}$ be the round that the strategy $s_i^{\star,t}$ of player i changes for the r-th time or her type changes while being allocated a resource. For any period $[\tau_{i,r}, \tau_{i,r+1})$ that the strategy $s_i^{\star,t}$ is fixed, the adaptive approximate regret property guarantees that the player's expected regret $R_i(\tau_r, \tau_{r+1}) = \mathbb{E}\Big[AdApxReg(s_i^{\star,1:T}, \epsilon, \tau_r, \tau_{r+1})\Big]$ with respect to this strategy is at most

$$R_i(\tau_r, \tau_{r+1}) \leq \frac{\log(NT)}{\epsilon}.$$

Moreover, if in period r, $x_i^t = \emptyset$, then by Property 1 we have that: $R_i(\tau_r, \tau_{r+1}) \leq 0$. Thus, if we denote with $X_{i,r}$ the indicator of whether in period r, $x_i^t = \emptyset$, we obtain:

$$R_i(\tau_r, \tau_{r+1}) \leq X_{i,r} \cdot \frac{\log(NT)}{\epsilon}.$$

Summing over the $K_i(x^{1:T})$ periods where strategies are fixed and summing over players, we can bound the total expected approximate regret across players by:

$$\mathbb{E}\left[\sum_i r_i^\star\right] \leq \mathbb{E}\left[\sum_{r=1}^{K_i(x^{1:T})} X_{i,r} \cdot \frac{\log(NT)}{\epsilon}\right] \leq \sum_i \mathbb{E}\left[K_i(x^{1:T})\right] \cdot \frac{\log(NT)}{\epsilon}$$

The rest of the proof follows the arguments of Proposition 6.2. More formally, using the notation there:

$$\begin{aligned}
\sum_t \mathbb{E}[U(s^t; v^t)] &= \frac{1}{1+\epsilon} \sum_t \sum_{i \in [n]} (1+\epsilon) \cdot \mathbb{E}[u_i(s^t; v_i^t)] \\
&\geq \frac{1}{1+\epsilon} \cdot \mathbb{E}\left[\sum_t \sum_i u_i(s_i^\star(v_i^t, x_i^t), s_{-i}^t; v_i^t) - \sum_i r_i^\star\right] \\
&\geq \frac{1}{1+\epsilon} \cdot \left(\lambda \sum_t \mathbb{E}[W(x^t; v^t)] - \mu \sum_t \mathbb{E}[\textsc{Rev}(s^t)] - \sum_i \frac{\mathbb{E}\left[K_i(x^{1:T})\right] \cdot \log(NT)}{\epsilon}\right)
\end{aligned}$$

The claimed bound comes again from rearranging terms, noting that $W(s^t; v) = U(s^t; v^t) + \textsc{Rev}(s^t)$, as well as applying that $\beta \cdot \mathbb{E}[W(x^t; v^t) \geq \mathbb{E}[\textsc{Opt}(v^t)]]$. ■

6.3 Classical smoothness analysis leads to ineffective results

In the previous section, we made two steps away from the classical smoothness analysis [116, 123]. First, we assumed that the learners apply adaptive approximate regret guarantees rather than the more classical no-regret against a benchmark fixed throughout time. Second, instead of taking the optimal solution as the benchmark allocation we wish to compete with, we turned to a β-approximately optimal solution. We now explain why both these deviations are essential to provide efficiency guarantees in dynamic environments.

Need for adaptive learning. From a player's perspective, using no-regret learning against a fixed comparator can be problematic. Consider a toy scenario where the are m items and a special player has value $\frac{1}{j}$ for item $j \in \{1, \ldots, \log(T)\}$. We assume that initially she has no competition but at time $\frac{i \cdot T}{m}$, player $i \in \{1, m-1\}$ arrives and this player significantly dominates her in value for all items and has the same preference order among items. As a result, after the i-th player arrives, our special player can realistically win only one of the i lower-valued items.

Any fixed benchmark provides a utility to the player of T/m (assuming that the payments are non-existing). This is because the i-th element is winnable only in the first $\frac{i \cdot T}{m}$ rounds. In fact, as we will see in the next chapter, classical algorithms such as vanilla multiplicative weights achieve performance exactly equal to that of the best fixed benchmark [59] and, if the learner uses something like that, then their utility will be therefore exactly T/m.

In contrast, adaptive learning guarantees that the learner can obtain reward from the best shifting benchmark: in this case selecting the i-th item in rounds $\left[\frac{(i-1) \cdot T}{m}, \frac{i \cdot T}{m} - 1\right]$. Since at these items, the learner faces essentially no competition, the reward she gets there is $H_m \cdot T/m$. Hence satisfying the adaptive approximate regret property leads to a multiplicative increase in performance of H_m at the expense of an additive dependence of m in the regret which vanishes as T grows large. Fortunately, the player can indeed achieve performance as good as that by many natural algorithms, which means that the stronger behavioral assumption is better suited to what players should and can aim in such evolving settings.

Not using the optimum as a benchmark in smoothness analysis. The original smoothness analysis uses the optimal allocation as the comparator x in the proof of Proposition 6.2; this gives the efficiency guarantee as $W(x; v) = \textsc{Opt}$. The issue with this approach in dynamic environments is that the optimal solution can be significantly unstable: a single change in a random player's type can significantly disturb the allocations of most other players. Since the number of these changes comes in the regret term, having such instability means that, unless the turnover probability is really small (the types of the players change very rarely), the regret term may dominate the efficiency term, making the bound vacuous.

To observe this instability, consider a simple setting with m items and $n = m$ players. Each player $i \in \{1, \ldots, m-1\}$ has value 1 for the items i and $i+1$ and 0 for all other items, while the m-th player has value 1 for item m. Clearly the optimal allocation assigns the i-th item to the i-th player. Now consider that a turnover happens to a random player and the new arriving player has value 1 only for item 1. If the player who turned over has identity higher than $m/2$ (this happens with probability half), the new optimal solution comes from an augmenting path affecting at least half of the players. This is because the new player gets item 1 and all players (until the player who turned over) switch to the next item. This means that in expectation at least $n/4$ players' allocations in the optimal solution is affected by a single type change in a random player. Hence the solution is unstable causing the additive error term in Theorem 6.3 to become really high (unless the probability of turnover is very small and hardly any changes occur).

In the next two sections, we describe how using approximately optimal solutions instead of the optimal solution can lead to significant improvement in stability of the resulting allocations. This implies that the efficiency guarantees we suggest are robust even to a large population churn (a constant fraction of the player set turning over) without significant degradation.

6.4 Robustness of efficiency in dynamic games via stability

The previous section demonstrates that, for the smoothness analysis to obtain efficient guarantees with dynamic population, there should exist a sequence of approximately optimal solutions that is relatively stable: a single change in a random player does not alter the allocation of most players. In this section, we demonstrate such a stable solution sequence for our main application (first-price auctions with unit-demand bidders) and subsequently instantiate the efficiency guarantee that this leads to. In the next section, we show a general way to achieve such stability via a connection to differential privacy.

Stability via layered greedy matching. To obtain a stable and approximately optimal solution sequence, we use a layered version of the greedy matching algorithm. The greedy algorithm initially does not allocate any item to any player. Subsequently it considers item valuations $v_i^t(j)$ in decreasing order assigning item j to player i if, when $v_i^t(j)$ is considered, neither item j nor player i are matched. To make this algorithm more stable we define the *layered*-greedy matching algorithm, which works as follows. Recall that $\rho > 0$ denotes the smallest non-zero value that a player has for any item. For a positive $\epsilon \leq 1/3$, we round each player's value down to the closest number of the form $\rho(1+\epsilon)^\ell$ for some integer ℓ, and run the greedy algorithm with these rounded values. It is well known that the greedy algorithm guarantees a solution that is within a factor of 2 to optimal. We lose an additional factor of $(1+\epsilon)$ by working with the rounded values.

The greedy algorithm will have many ties and we will resolve ties in a way to make the output stable. In particular, among all player-item pairs with the same rounded value at round t, we break ties in favor of pairs matched in the previous round $t-1$. Note that this neither affects anything in the mechanism nor makes assumption on how the mechanism breaks ties; it is just inside the proof to show the existence of an approximately optimal stable solution sequence.

We now provide the key ingredient for our efficiency guarantee, a lemma showing that the above algorithm provides a sequence of near-optimal and stable solutions; hence such a sequence exists for the underlying optimization problem.

Lemma 6.1 (Stability via layered-greedy matching)**.** For any $\epsilon > 0$, there exists a randomized sequence of solutions for the underlying matching problem (depending on the randomness in the type sequence) such that a) the total welfare at any round is near-optimal: $W(x^t; v^t) \geq \frac{1-\epsilon}{2}\textsc{Opt}(v^t)$ and b) the expected number of changes in the allocation of a player is bounded by

$$\mathbb{E}\left[\sum_i K_i(x^{1:T})\right] \leq m \cdot (2 + 2pT) \cdot \log_{(1+\epsilon)}(1/\rho).$$

Proof. The solutions in the sequence of the theorem come by applying the layered-greedy matching algorithm with parameter $\epsilon > 0$. The approximation result holds as we lose a factor of 2 due to the greedy algorithm and another factor of $(1+\epsilon)$ due to the layers. Since $\frac{1}{1+\epsilon} > 1-\epsilon$, result (a) follows.

To show the stability let $\ell(v_i^t(j))$ be the highest integer ℓ such that $v_i^t(j) \geq \rho(1+\epsilon)^{\ell-1}$, i.e. the rounded down version of $v_i^t(j)$ is $\rho(1+\epsilon)^{\ell(v_i(j))-1}$, which we call the layer of this value. For example, any value in the range $[\rho, \rho(1+\epsilon))$ is in layer 1. Let $\ell^t(j)$ denote $\ell(v_i^t(j))$ if item j is assigned to player i at time t, and let $\ell^t(j) = 0$ if item j is not assigned at time t. We will use the potential function

$$\Phi(x^t) = \sum_j \ell^t(j)$$

to show stability. As all values are upped bounded by 1, the number of possible values that the potential function can take is $m \cdot \log_{(1+\epsilon)}(1/\rho)$.

The crux of the proof relies in the following steps. First, we show that changes in assignments of non-departing players correspond to increases in the potential function. Moreover, the potential function can decrease only due to departures: when a player assigned to item j leaves at time t, this immediately decreases the potential function by $\ell^t(j) \leq \log_{(1+\epsilon)}(1/\rho)$. Ignoring initial steps, the aggregate increase in the potential function is the same as the aggregate decrease. Hence, the expected number of changes is upper bounded by bounding the expected decrease in the potential function. This argument is formalized below.

The allocation of the solution can change only due to a turnover: either a player that holds an item in the greedy solution departs and leaves her previously assigned item open for current players, or a new player arrives and gets assigned to an item. Every time that a player that holds an item departs, she leaves that item temporarily free in the greedy solution. Unless this is the last time that the item has a holder in the greedy solution, at some point (either at the same round or in the future), some player i will get assigned to this item. Due to the layered version of the greedy algorithm needs to increase the potential function by at least 1. If player i was previously assigned to another item, this item becomes free and some other player may move to that item by increasing again the potential function by at least 1. As a result, the total increase in the potential function, caused by someone getting an unassigned item is associated to at most 2 changes. The case where the item remains unassigned for all the future rounds, contributes in total at most m extra changes in allocation (over the whole time horizon).

Now consider the scenario where an arriving player misplaces another player from the greedy solution (instead of getting an unassigned item). For this to happen, her rounded value is higher than the one of the current owner; hence, the potential function increases by 1. This affects the allocation of the previous holder of the item who may either cease being allocated or replace a player in another item again increasing the potential function by at least 1 (and this may propagate across subsequently misplaced players as well). As a result, each increase in the potential function that is caused by someone getting a previously assigned item contributes at most 2 changes in the allocation of other players.

Combined with the previous point, the total number of changes is at most:

$$\sum_i K_i(x^{1:T}) \le 2 \cdot \text{Total Increase in } \Phi + m \tag{6.2}$$

Since the potential function can only get integer non-negative values and is bounded by $m \cdot \log_{(1+\epsilon)}(1/\rho)$ and taking into account end-game effects, the total increase in the potential function is:

$$\text{Total Increase in } \Phi \le \text{Total Decrease in } \Phi + m \cdot \log_{(1+\epsilon)}(1/\rho). \tag{6.3}$$

We are therefore left to bound the expected total decrease in the potential function. This can only happen when a player among the ones that hold items departs. Each such player departs with probability p and hence the expected number of such players departing at each round is at most $m \cdot p$ (which is independent of the number of players and depends only on the number of items). Whenever this happens, the potential function can decrease by at most $\log_{(1+\epsilon)}(1/\rho)$ since this is the maximum layer the corresponding item can be in. As a result, the expected decrease in the potential function is at most:

$$\mathbb{E}[\text{Total Decrease in } \Phi] \le p \cdot m \cdot T \cdot \log_{(1+\epsilon)}(1/\rho) \tag{6.4}$$

The theorem follows by combining (6.2), (6.3), and (6.4). ■

Smoothness with discrete bidding spaces. To apply Theorem 6.3, we need to establish that the mechanism with the discrete bidding space is smooth. $(1/2, 1)$-smoothness of the simultaneous first price auction with submodular bidders (a super-set of unit-demand valuations) and continuous bids was known by [123]. A simple modification of the result of [123] shows that if the discretization is fine enough, then the mechanism is approximately $(1/2, 1)$ solution-based smooth. Since the techniques are similar to [123], we defer this proof to Appendix E.

Lemma 6.2 (Smoothness of simultaneous discrete-bidding first price auction)**.** The simultaneous first price mechanism where players are restricted to bid on at most d items and on each item submit a bid that is a multiple of $\delta \cdot \rho$, is a solution-based $\left(\frac{1}{2} - \delta, 1\right)$-smooth mechanism, when players have submodular valuations, such that all marginals are either 0 or at least ρ and such that each player wants at most d items, i.e. $v_i^t(S) = \max_{T \subseteq S : |T| = d} v_i^t(T)$.

Efficiency guarantee We now provide the efficiency guarantee that has many nice properties. First, it is parametric; we get an extra factor of 2 due to the use of greedy algorithm in the proof but, other than that, the additional error goes to 0 as $p \to 0$. Second, the turnover probability can be very high without big loss in efficiency as there is no dependence on the number of players or items, depends only the range of item valuations. In particular, even if a constant fraction of the players is changing at every round then we still do not see much loss in efficiency, which makes the guarantee meaningful even with high player turnover.

Theorem 6.4 (Main theorem for unit-demand bidders)**.** Consider simultaneous first price auctions with dynamic population, non-overbidding unit-demand bidders, and discrete bidding space of multiples of $\delta \cdot \rho$ for some $\delta > 0$. Assume that the average optimal welfare in each round is at least $m\rho$, that is

$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[\textsc{Opt}(v^t)] \geq m\rho$ (all items can be allocated for the minimum value). If players satisfy the adaptive approximate regret property and $T \geq \frac{1}{p}$ then $\forall \epsilon' > 0$:

$$\textstyle\sum_t \mathbb{E}[W(s^t; v^t)] \geq \left(\frac{(1-2\delta)\cdot(1-\epsilon')}{4\cdot(1+\epsilon)} - p\cdot\left(\frac{4\log_{(1+\epsilon')}(1/\rho)\ln(NT)}{\rho}\right)\right)\mathbb{E}[\textsc{Opt}(v^t)]$$

where N is the number of different strategies considered by a player.

Proof. We apply Theorem 6.3 with x^t being the allocation of the layered-greedy algorithm with parameter $\epsilon' > 0$. We use the $(1/2-\delta, 1)$ solution-based smoothness of the first price auction with discrete bidding space established in Lemma 6.2 and the stability of the solution sequence produced by the layered-greedy algorithm establised in Lemma 6.1. Using that $pT > 1$, we obtain:

$$\sum_t \mathbb{E}[W(s^t; v^t)] \geq \frac{(1-2\delta)(1-\epsilon')}{4\cdot(1+\epsilon)}\sum_t \mathbb{E}[\textsc{Opt}(\vec{v}^t)] - m\cdot 4\cdot p\cdot T\cdot \log_{(1+\epsilon')}(1/\rho)\cdot \ln(NT)$$

The theorem then follows using the lower bound on the optimum solution $\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[\textsc{Opt}(\vec{v}^t)] \geq m\rho$. Since we never run the greedy-layered matching, ϵ' is just inside the proof; it only affects the solution x^t and not the possible bids, and therefore it is fine to define this parameter arbitrarily. ■

6.5 Stability in dynamic games via differential privacy

In the last technical section of this chapter, we provide a general way to derive stable approximately optimal solutions (the essential buliding block in the efficiency guarantees), by stressing a connection of such solutions to differential privacy. Differential privacy offers a general framework to find solutions that are close to optimal, yet more stable to changes in the input than the optimum itself. To guarantee privacy, the output of the algorithm should depend only minimally on any single player's input. This is exactly what we need in our framework.

Background on differential privacy. Differential privacy has been developed for databases storing private information for a population. A database $D \in \mathbf{V}^n$ is a vector of inputs, one for each user. Two databases are *i-neighbors* if they differ just in the i-th coordinate, i.e. only in the input of the i-th user. If two databases are i-neighbors for some i, they are called *neighboring databases*. Dwork et al. [50] define an algorithm as *differentially private* if one user's information has little influence on the outcome. In the setting of a game or mechanism the outcome for player i clearly should depend on player i's input (her claimed valuation, or source destination pair), so cannot be differentially private. The notion of *joint differential privacy* has been developed by Kearns et al. [76] to adapt differential privacy to such settings. We use $\mathbf{X}$ to denote the set of possible outcomes for one player, so an algorithm in this context is a randomized mapping $\mathcal{A} : \mathbf{V}^n \to \mathbf{X}^n$. The algorithm is jointly differentially private, if for all players i, the output for all other players is differentially private in the input of player i. More formally:

Definition 6.4 ([76]). *An algorithm $\mathcal{A} : \mathbf{V}^n \to \mathbf{X}^n$ is (ϵ, δ)- jointly differentially private if for every i, every pair of i-neighbors $D, D' \in \mathbf{V}^n$, and every subset of outputs $S \subseteq \mathbf{X}^{n-1}$.*

$$\mathbb{P}[\mathcal{A}(D)_{-i} \in S] \leq e^{\epsilon} \cdot \mathbb{P}[\mathcal{A}(D')_{-i} \in S] + \delta \tag{6.5}$$

If $\delta = 0$, we say that $\mathcal{A}$ is ϵ-jointly differentially private.

Over the last years there have been a number of algorithms developed that solve problems near-optimally in a differentially private way; see the book of Dwork and Roth [48] for a survey. Via connecting joint different privacy to stable solution sequences, we provide dynamic efficiency guarantees for settings like online advertising and routing via taking advantage of algorithms for solving matching problems [70] and finding socially optimal routing [115] respectively.

Stability via differential privacy. We now show that differentially private solutions imply stability. This is the main building block for efficiency guarantees.

Lemma 6.3. Suppose that there exists an algorithm $\mathcal{A} : \mathbf{V}^n \to \Delta(\mathbf{X}^n)$ that is (ϵ, δ)-jointly differentially private, takes as input a valuation profile v and outputs a distribution of solutions such that a sample from this distribution is feasible with probability $1 - \gamma$, and is β-approximately optimal in expectation (for $0 \le \epsilon \le 1/2$, $\beta > 1$, $\delta > 0$, and $0 < \gamma < 1$). Consider a sample $v^{1:T}$ from the distribution of valuations produced by the adversary in a repeated mechanism with dynamic population $\mathcal{M} = (M, p, T)$. There exists a randomized sequence of solutions $x^{1:T}$ for the sequence $v^{1:T}$, such that a) for each $1 \le t \le T$, x^t conditional on v^t is a β-approximation to $\textsc{Opt}(v^t)$ in expectation over the randomness of the algorithm: $\beta \cdot \mathbb{E}[W(x^t; v^t)] \le \mathbb{E}[\textsc{Opt}(v^t)]$ and b) the expected number of changes in the solution is bounded by

$$\mathbb{E}\left[\sum_i K_i(x^{1:T})\right] \le p \cdot n \cdot T \cdot (1 + n(2\epsilon + 2\gamma + \delta))$$

To prove this main lemma, we require two auxiliary lemmas. First, we bound the total variation distance between the outputs of a differentially private algorithm on two inputs that differ only on one coordinate (the type of one player). Total variation distance is a general measure for the distance between distributions. For two distributions μ and η on some finite probability space Ω the following are two equivalent versions of the total variation distance:

$$d_{tv}(\mu, \eta) = \frac{1}{2}\|\mu - \eta\|_1 = \max_{A \subset \Omega}(\mu(A) - \eta(A)), \tag{6.6}$$

where in the 1-norm in the middle we think of μ and η as a vector of probabilities over the possible outcomes.

Lemma 6.4. Suppose that $\mathcal{A} : \mathbf{V}^n \to \Delta(\mathbf{X}^n)$ is an (ϵ, δ)-joint differentially private algorithm with failure probability γ (for $0 \le \epsilon \le 1/2$, $\delta > 0$, and $0 < \gamma < 1$)

that takes as input a valuation profile v and outputs a distribution over feasible solutions σ. Let σ and σ' be the algorithm's outputs on two inputs v and v' that differ only in coordinate i. Then we can bound the total variation distance between σ_{-i} and σ'_{-i} by $d_{tv}(\sigma_{-i}, \sigma'_{-i}) \leq (2\epsilon + \delta)$.

Proof. Condition (6.5) of joint differential privacy guarantees that if we let $S \subseteq \mathbf{X}^n_{-i}$ be a subset of possible solutions for players other than i, and $\sigma_{-i}(S)$ and $\sigma'_{-i}(S)$ be the probability that the two distributions assign on S, then for any S: $\sigma_{-i}(S) \leq e^{\epsilon} \cdot \sigma'_{-i}(S) + \delta$. Since $\epsilon \leq 1/2$, we can use the bound $e^{\epsilon} \leq 1 + 2\epsilon$ to get that $\sigma_{-i}(S) - \sigma'_{-i}(S) \leq 2\epsilon\sigma'_{-i}(S) + \delta \leq 2\epsilon + \delta$. Thus by the second definition of the total variation distance in Equation (6.6) we get that $d_{tv}(\sigma_{-i}, \sigma'_{-i}) \leq 2\epsilon + \delta$. ■

Second, we use a simple lemma from basic probability theory.

Lemma 6.5 (Coupling Lemma)**.** Let μ and η be two probability measures over a finite set Ω. There is a coupling ω of (μ, η), such that if the random variable (X, Y) is distributed according to ω, then the marginal distribution on X is μ, the marginal distribution on Y is η, and

$$\mathbb{P}[X \neq Y] = d_{tv}(\mu, \eta),$$

Proof of Lemma 6.3. Suppose that $\mathcal{A} : \mathbf{V}^n \to \Delta(\mathbf{X}^n)$ is an (ϵ, δ)-joint differentially private algorithm as described in the definition of the lemma. The differentially private algorithm fails with probability γ. We denote with σ the output distribution over solutions for an input v, where we use the optimal solution in the low probability event that the algorithm fails. (Equivalently $\mathcal{A}$ could be a randomized algorithm and σ its implicit distribution over solutions).

Let $\sigma^1, \ldots, \sigma^T$, be the sequence of distributions output by $\mathcal{A}$ when run on a deterministic sequence of valuation profiles $v^1, \ldots, v^T$ with the modification described in the paragraph above. To simplify the discussion we assume that only one player changes valuation at each time-step t. Essentially we break every transition from time-step t to $t+1$ into many sequential transitions where only one player changes at every time step, and then deleting the solutions from the resulting sequence that correspond to the added steps. Thus the number of steps within this proof should be thought as being equal to $n \cdot p \cdot T$ in expectation.

By Lemma 6.4, we know that the total variation distance of two consecutive distributions without the modification of replacing failures with the optimal solution is at most $2\epsilon + \delta$. Since, by the union bound, the probability that any of the two consecutive runs of the algorithm fail is at most 2γ, we can show that the total variation distance of the latter modified output is at most $2\epsilon + \delta + 2\gamma$, i.e. for any $t \in [T]$: $d_{tv}(\sigma_{-i}^{t+1}, \sigma_{-i}^{t}) \leq 2\epsilon + \delta + 2\gamma$ (see Lemma 6.6 for a formal proof).

We can turn the sequence of distributions $\sigma^1, \ldots, \sigma^T$ into a distribution of sequences of allocations $x^{1:T}$ by coupling the randomness used to select the solutions in different distributions σ^t. To do this, we take advantage of the coupling lemma from probability theory (Lemma 6.5). If at step t no player changes values, then $\sigma^t = \sigma^{t+1}$, and we select the same outcome from the two distributions, so we get $\mathbb{P}\left[x_{-i}^t \neq x_{-i}^{t+1}\right] = 0$.

Now consider a step in which a player i changes her private type v_i. We use Lemma 6.5 to couple x_{-i}^{t+1} and x_{-i}^t so that[3]

$$\mathbb{P}[x_{-i}^{t+1} \neq x_{-i}^t] = d_{tv}(\sigma_{-i}^{t+1}, \sigma_{-i}^t) \leq 2\epsilon + \delta + 2\gamma. \tag{6.7}$$

[3]One can think of it as sampling x^{t+1} conditional on x^t and assuming the joint distribution of x^t and x^{t+1} is as prescribed by the coupling lemma applied to σ^t and σ^{t+1}. This is to address concerns that x^t is already coupled with x^{t-1} in the previous step.

Note that this couples the ith coordinate x_i^{t+1} and x_i^t in an arbitrary manner, which is fine, as we assumed that the valuation of player i changes at this step.

We have defined a probability distribution of sequences $x^{1:T}$ for every fixed sequence of valuations $v^{1:T}$. We extend this definition to random sequences of valuation in the natural way adding the distribution of valuations $v^{1:T}$.

We claim that the resulting random sequences of (valuation,solution) pairs satisfies the statement of the theorem: the β-approximation follows by the guarantees of the private algorithm and by the fact that we use the optimal solution when the algorithm fails. Next we argue about the stability of the sequence. Consider a player i, and the distribution of her sequence $(v_i^{1:T}, x_i^{1:T})$. In each step t her valuation v_i^t changes with probability p, contributing pT in expectation to the number of changes. In a step t when some other value $j \neq i$ changes, we use (6.7) to bound the probability that $x_i^t \neq x_i^{t+1}$ by $2\epsilon + \delta + 2\gamma$. Thus any change in the value of some other player j contributes at most $(2\epsilon + 2\gamma + \delta)$ to the expectation of the number of changes for player i. The expected number of such changes in other values is $(n-1)pT$ over the sequence, showing that

$$\mathbb{E}[K_i] = pT + (n-1)pT(2\epsilon + 2\gamma + \delta) \leq pT(1 + n(2\epsilon + 2\gamma + \delta)).$$

Summing over players, we obtain the lemma. ■

Lemma 6.6. Let q and q' be the output of an (ϵ, δ)-joint differentially private algorithm with failure probability γ, on two valuation profiles v and v' that differ only in coordinate i. Let σ and σ' be the modified output where the outcome is replaced with optimal outcome when the algorithm fails. Then:

$$d_{tv}(\sigma, \sigma') \leq 2\epsilon + \delta + 2\gamma$$

Proof. Consider two random coupled random variables y, y' that are implied by

Lemma 6.5 applied to distributions q and q', such that $y \sim q$ and $y' \sim q'$ and $\mathbb{P}[y \neq y'] = d_{tv}(q, q') \leq 2\epsilon + \delta$ (by Lemma 6.4). Now consider two other random variables x and x' where $x = y$ except for the cases where y is an outcome of a failure in which case x is equal to the welfare optimal outcome and similarly for x' and y'. Obviously: $x \sim \sigma$ and $x' \sim \sigma'$, thus (x, x') is a valid coupling for distributions σ and σ'. Thus if we show that $Pr[x \neq x'] \leq 2\epsilon + \delta + 2\gamma$, then by properties of total variation distance $d_{tv}(\sigma, \sigma') \leq Pr[x \neq x'] \leq 2\epsilon + \delta + 2\gamma$, which is the property we want to show.

Let fail be the event that either y or y' is the outcome of a failed run of the algorithm. Then by the union bound $\mathbb{P}[\text{fail}] \leq 2\gamma$. Thus we have:

$$\begin{aligned}
\mathbb{P}[x \neq x'] = &\ \mathbb{P}[x \neq x' \mid \neg\text{fail}] \cdot \mathbb{P}[\neg\text{fail}] + \mathbb{P}[x \neq x' \mid \text{fail}] \cdot \mathbb{P}[\text{fail}] \\
\leq &\ \mathbb{P}[x \neq x' \mid \neg\text{fail}] \cdot \mathbb{P}[\neg\text{fail}] + 2\gamma \\
= &\ \mathbb{P}[y \neq y' \mid \neg\text{fail}] \cdot \mathbb{P}[\neg\text{fail}] + 2\gamma \\
\leq &\ \mathbb{P}[y \neq y'] + 2\gamma \leq d_{tv}(q, q') + 2\gamma \leq 2\epsilon + \delta + 2\gamma
\end{aligned}$$

This completes the proof of the Lemma. ■

Efficiency guarantee for dynamic games via differential privacy. We can now provide the resulting efficiency guarantee.

Theorem 6.5. Consider a repeated mechanism with dynamic population $\mathcal{M} = (M, T, p)$, such that the stage mechanism M is solution-based (λ, μ)-smooth and $T \geq \frac{1}{p}$. Assume that there exists an (ϵ, δ)-joint differentially private algorithm $\mathcal{A} : \mathbf{V}^n \to \mathbf{X}^n$ with error parameter γ that satisfies the conditions of Lemma 6.3. If players satisfy the adaptive approximate regret property then:

$$\sum_t \mathbb{E}[W(s^t; v^t)] \geq \frac{\lambda}{\beta \max(\mu, 1+\epsilon)} \sum_t \mathbb{E}[\textsc{Opt}(v^t)] - \sum_i \frac{2pnT(1 + n(\epsilon + \gamma + \delta)) \log(NT)}{\max(\mu, 1+\epsilon) \cdot \epsilon}$$

Proof. The proof follows directly by combining Theorem 6.3 and Lemma 6.3. ■

Using this theorem, we can obtain efficiency guarantees for simultaneous auctions where players have submodular valuation functions, assuming the number of players is large enough. This comes from using the algorithm *PAlloc* of Hsu et al. [70] which provide near-optimal differentially private guarantees in large markets. Similarly, applying the equivalent of the above theorem for cost games (which is omitted from this thesis), we can provide efficiency guarantees for routing games via the differenitally private algorthm of Rogers et al. [115].

6.6 Remarks

More information about the papers. The results in this chapter are based mostly on joint work with Vasilis Syrgkanis and Éva Tardos [94]. The connection between approximate regret learners and efficiency of outcomes is based on joint work with Dylan Foster, Zhiyuan Li, Karthik Sridharan, and Éva Tardos [53]. The dynamic population model, the interval-based behavioral assumption, and the stability results leading to efficiency appear in [94]. In that paper, we also instantiate the differential privacy framework to simultaneous auctions with submodular valuations and routing games as hinted in the end of the last section.

Learning as a behavioral assumption. Decentralized dynamics as a model of player behavior in repeated settings dates back to the seminal work of Brown on fictitious play in two-player zero-sum games [31] which converges to the so called min-max value of the game [114]. The rate of this convergence has been subsequently extensively studied both for fictitious play [114, 45] as well as for other learning dynamics in zero-sum games [44, 111]. Hart and Mas Collel showed that in more general games, uncoupled dynamics do not converge to Nash equilibria but provided a dynamic satisfying a stronger notion of regret (internal regret) that converges to the so called correlated equilibria. Despite that, Blum et al. [27] showed that in games such as routing, generic decentralized no-regret learning dynamics do converge to an approximate form of Nash equilibria. Subsequently, multiple works have aimed to understand the exact topological trajectories of decentralized dynamics to shed more light on their convergence, for example [79, 98].

In this chapter, we focus on properties of the outcomes in game settings where players use generic no-regret learning dynamics instead of analyzing the exact dynamic. In particular, we analyze the social welfare in this repeated game where the behavioral assumption for the players is that they use no-regret learning. Quantifying the inefficiency caused by the selfish behavior of players is due to the seminal works on price of anarchy by Koutsoupias and Papadimitriou [82], and Roughgarden and Tardos [117], with Nash equilibrium as a notion for selfish behavior. Using the learning behavioral assumption to capture the selfish behavior of the players in this context was initiated by Blum et al. [29]. The smoothness extension theorem of Roughgarden [116] provided a general framework to make price of anarchy guarantees hold even under this weaker learning behavioral assumption. Our work goes one step further and provides a framework to make these results robust to drastically evolving game settings.

CHAPTER 7

A FAIRNESS VIEW ON ONLINE LEARNING

In the final facet of this thesis, we expand the focus of online decision-making beyond a mere optimization perspective and aim to address the greater societal context of these decisions. Until now, we focused on understanding how online decision-making techniques can help platforms and other agents adapt to the optimization complexities of modern systems. Recently, there are emerging concerns regarding approaches that focus on just optimizing objectives such as revenue or welfare. Ethical considerations slowly start coming to the picture with respect to the functioning of online markets as platforms' decisions affect multiple different agents. Hence there is a sense of urgency to undertand shortcomings with respect to important considerations such as fairness and privacy, and mitigate them through either algorithmic or regulatory interventions.

One consideration that has become more and more prominent with respect to how algorithmic decisions affect people is fairness across different population groups. Although there is still ongoing debate on what classifies as *fair* [19, 24, 41], recent works identify morally objectionable practices [46, 7, 8] suggest operational remedies for particular tasks [49, 65, 71] or point to fundamental obstacles preventing them [40, 78]. However, most of them focus on analyzing existing datasets in an offline manner which disregards the fact that decisions often need to be made in an online manner without the benefit of the complete dataset. Approaches that incorporate the online nature of modern markets tend to heavily rely on the input being completely i.i.d. (see Section 7.6). This then enables them to imitate the offline approaches by initially exploring to find the best *fair* policy (for the fairness notion of interest) and then repeatedly use it.

In this chapter, we focus on understanding the extra complexities that the online aspect of this decision-making adds to the picture in settings where the i.i.d. assumption on the input is not necessarily valid (e.g. because the input evolves as we discussed in the previous chapter). In particular, for these settings we wish to understand the interplay between online optimization and fairness with respect to different notions of group fairness. Our goal is to identify which group fairness notions are compatible with optimization occurring in an online manner and point to places where system designers may need to be extra thoughtful when dealing with non-i.i.d. online datasets.

7.1 Preliminaries on online learing with multiple groups

Before discussing fairness considerations, we first formally describe the online learning setting with multiple groups; this slightly extends the exposition in Chapter 2 but all notions are redefined for completeness and notational simplicity. In particular, unlike that chapter where the losses were treated as abstract, here we provide more context on their origin, in order to distinguish between different types of mistakes (e.g. false positives and false negatives, defined in Section 7.2).

Online learning setting with group context. We focus on the full-feedback adversarial online learning setting (also referred to as learning with expert advice). A learner needs to make sequential decisions for T rounds by combining the predictions of a finite set $\mathcal{F}$ of d hypotheses (also referred to as *experts*). We denote the outcome space by $\mathcal{Y}$; in binary classification, this corresponds to $\mathcal{Y} = \{+, -\}$. Additionally, we introduce a set of disjoint groups by $\mathcal{G}$ which identifies subsets of the population based on a protected attribute (gender, ethnicity, income, etc).

The *online learning setting with group context* proceeds in T rounds. Each round t is associated with a group context $g(t) \in \mathcal{G}$ and an outcome $y(t) \in \mathcal{Y}$. We denote the resulting T-length time-group-outcome sequence tuple by $\sigma = \{(t, g(t), y(t)) \in \mathbb{N} \times \mathcal{G} \times \mathcal{Y}\}_{t=1}^{T}$. This is a random variable that can depend on the randomness in the generation of the groups and the outcomes. We use the shorthand $\sigma^{1:\tau} = \{(t, g(t), y(t)) \in \mathbb{N} \times \mathcal{G} \times \mathcal{Y}\}_{t=1}^{\tau}$ to denote the subsequence until round τ. The exact protocol for generating these sequences is described below. At round $t = 1, 2, \ldots, T$:

1. An example with group context $g(t) \in \mathcal{G}$ either arrives stochastically or is adversarially selected.
2. The learning algorithm or *learner* $\mathcal{L}$ commits to a probability distribution $p^t \in \Delta(d)$ across experts where p_f^t denotes the probability that she follows the advice of expert $f \in \mathcal{F}$ at round t. This distribution p^t can be a function of the sequence $\sigma^{1:t-1}$. We call the learner *group-unaware* if she ignores the group context $g(\tau)$ for all $\tau \leq t$ when selecting p^t.
3. An adversary $\mathcal{A}$ then selects an outcome $y(t) \in \mathcal{Y}$. The adversary is called *adaptive* if the groups/outcomes at round $t = \tau + 1$ are a function of the realization of $\sigma^{1:\tau}$; otherwise she is called *oblivious*. The adversary always has access to the learning algorithm, but an adaptive adversary additionally has access to the realized $\sigma^{1:t-1}$ and hence also knows p^t.

   Simultaneously, each expert $f \in \mathcal{F}$ makes a prediction $\hat{y}_f^t \in \hat{\mathcal{Y}}$, where $\hat{\mathcal{Y}}$ is a generic prediction space. For example, in binary classification, the prediction space could simply be the positive or negative labels: $\hat{\mathcal{Y}} = \{+, -\}$, or the probabilistic score: $\hat{\mathcal{Y}} = [0, 1]$ with $\hat{y}_f^t$ interpreted as the probability the expert $f \in \mathcal{F}$ assigns to the positive label in round t, or even an uncalibrated score like the output of a support vector machine: $\hat{\mathcal{Y}} = \mathbb{R}$.

Let $\ell : \hat{\mathcal{Y}} \times \mathcal{Y} \to [0,1]$ be the loss function between predictions and outcomes. This leads to a corresponding loss vector $\ell^t \in [0,1]^d$ where $\ell^t_f = \ell\big(\hat{y}^t_f, y(t)\big)$ denotes the loss the learner incurs if she follows expert $f \in \mathcal{F}$.

4. The learner then observes the entire loss vector ℓ^t (full feedback) and incurs expected loss $\sum_{f \in \mathcal{F}} p^t_f \ell^t_f$. For classification, this feedback is obtained by observing $y(t)$.

In this chapter, we consider a setting where all the experts $f \in \mathcal{F}$ are fair in isolation (formalized below). Regarding the group contexts, our main impossibility results (Theorems 7.1 and 7.2) assume that the group contexts $g(t)$ arrive stochastically from a fixed distribution, while our positive result (Theorem 7.3) holds even when they are adversarially selected.

For simplicity of notation, we assume throughout the presentation that the learner's algorithm is producing the distribution p^t of round $t = \tau + 1$ deterministically based on $\sigma^{1:\tau}$ and therefore all our expectations are taken only over σ which is the case in most algorithms. Our results extend when the algorithm uses extra randomness to select the distribution.

Regret notions. The typical way to evaluate the performance of an algorithm in online learning is via the notion of *regret*. This has already been discussed in Chapter 2 but we redefine it here comparing the expected performance of the algorithm to the one of the best expert in hindsight on the realized sequence σ.

$$Regret_T = \sum_{t=1}^{T} \sum_{f \in \mathcal{F}} p^t_f \ell^t_f - \min_{f^\star \in \mathcal{F}} \sum_{t=1}^{T} \ell^t_{f^\star}.$$

For any fixed loss vector (determined by the groups and outcomes at every round), the above definition is the same as what we called expected regret in Chapter 2. In our impossibility results, we assume that the losses come in fact from particular distributions and we are interested in the expectation of this regret notion with respect to the randomness in the groups and outcomes (as we want to argue that even in expectation, the vanishing regret property is not compatible with particular fairness notions). To facilitate exposition, we therefore incorporate the algorithm's expectation inside the regret notion in the above definition; hence, any additional expectation relates to the groups and outcomes.

An algorithm satisfies the no-regret property (or Hannan consistency) in our setting if for any losses realizable by the above protocol, the regret is sublinear in the time horizon T, i.e. $Regret_T = o(T)$. This property ensures that, as time goes by, the average regret vanishes. Many online learning algorithms, such as multiplicative weights updates satisfy this property with $Regret_T = O(\sqrt{T \log(d)})$.

We focus on the notion of *approximate regret* (as in Chapters 2 and 6), which is a relaxation of regret that gives a small multiplicative slack to the algorithm. More formally, ϵ-approximate regret with respect to expert $f^\star \in \mathcal{F}$ is defined as:

$$ApxReg_{\epsilon,T}(f^\star) = \sum_{t=1}^{T} \sum_{f \in \mathcal{F}} p_f^t \ell_f^t - (1+\epsilon) \sum_{t=1}^{T} \ell_{f^\star}^t.$$

We note that typical algorithms guarantee $ApxReg_{\epsilon,T}(f^\star) = O(\ln(d)/\epsilon)$ simultaneously for all experts $f^\star \in \mathcal{F}$. When the time-horizon is known in advance, by setting $\epsilon = \sqrt{\ln(d)/T}$, such a bound implies the aforementioned regret guarantee. In the case when the time horizon is not known, one can also obtain a similar guarantee by adjusting the learning rate of the algorithm appropriately.

7.2 On combining fair expert advice fairly

Group fairness in online learning. We now define non-discrimination (or fairness) with respect to a particular evaluation metric $\mathcal{M}$, e.g. in classification, the false negative rate metric (FNR) is the fraction of examples with positive outcome that the algorithm predicts as negative incorrectly. For any realization of the time-group-outcome sequence σ and any group $g \in \mathcal{G}$, metric $\mathcal{M}$ induces a subset of the population $\mathcal{S}_g^\sigma(\mathcal{M})$ that is relevant to it. For example, in classification, $\mathcal{S}_g^\sigma(FNR) = \{t : g(t) = g, y(t) = +\}$ is the set of positive examples of group g. The performance of expert $f \in \mathcal{F}$ on the subpopulation $S_g^\sigma(\mathcal{M})$ is denoted by

$$\mathcal{M}_f^\sigma(g) = \frac{1}{|\mathcal{S}_g^\sigma(\mathcal{M})|} \sum_{t \in \mathcal{S}_g^\sigma(\mathcal{M})} \ell_f^t.$$

Definition 7.1. *An expert $f \in \mathcal{F}$ is called* ***fair in isolation with respect to metric*** *$\mathcal{M}$ if, for every sequence σ, her performance with respect to $\mathcal{M}$ is deterministically the same across groups, i.e. $\mathcal{M}_f^\sigma(g) = \mathcal{M}_f^\sigma(g')$ for all $g, g' \in \mathcal{G}$.*

Similarly, the learner's performance on this subpopulation is

$$\mathcal{M}_{\mathcal{L}}^\sigma(g) = \frac{1}{|\mathcal{S}_g^\sigma(\mathcal{M})|} \sum_{t \in \mathcal{S}_g^\sigma(\mathcal{M})} \sum_{f \in \mathcal{F}} p_f^t \ell_f^t.$$

To formalize our non-discrimination desiderata, we require the algorithm to have similar expected performance across groups, when given access to fair in isolation predictors. We make the following assumptions to avoid trivial impossibility results due to low-probability events or underrepresented populations. First, we take expectation over sequences generated by the adversary $\mathcal{A}$ (that has access to the learning algorithm $\mathcal{L}$). Second, we require the relevant subpopulations to be, in expectation, *large enough*. Our positive results do not depend on either of these assumptions. More formally:

Definition 7.2. *Consider a set of experts $\mathcal{F}$ such that each expert is fair in isolation with respect to metric $\mathcal{M}$. Learner $\mathcal{L}$ is called α-**fair in composition with respect to metric** $\mathcal{M}$ if, for all adversaries that produce $\mathbb{E}_\sigma[\min(|S_g^\sigma(\mathcal{M})|, |S_{g'}^\sigma(\mathcal{M})|)] = \Omega(T)$ for all $g, g' \in \mathcal{G}$, it holds that:*

$$\left|\mathbb{E}_\sigma\left[\mathcal{M}_{\mathcal{L}}^\sigma(g)\right] - \mathbb{E}_\sigma\left[\mathcal{M}_{\mathcal{L}}^\sigma(g')\right]\right| \leq \alpha.$$

We note that, in many settings, we wish to have non-discrimination with respect to multiple metrics simultaneously. For instance, the notion of equalized odds [65] requires fairness in composition both with respect to false negative rate and with respect to false positive rate (defined analogously). Since we provide an impossibility result for equalized odds, focusing on only one metric makes the result even stronger.

Optimizing performance vs preserving fairness. Our goal is to develop online learning algorithms that combine fair in isolation experts in order to achieve both vanishing average expected ϵ-approximate regret, i.e. for any fixed $\epsilon > 0$ and $f^\star \in \mathcal{F}$, $\mathbb{E}_\sigma\left[ApxReg_{\epsilon,T}(f^\star)\right] = o(T)$, and also non-discrimination with respect to fairness metrics of interest. We show that, when the fairness notion requires to balance false negative rates across groups, satisfying both the aforementioned guarantees is not possible (Section 7.3). In contrast, we can design effective algorithms that satisfy an alternate fairness notion: balance of the average performance experienced by each group (Section 7.4). Crucially, this requires a specific property that prevents the algorithm from being very good; at the absence of this property, impossibility results come back even for the latter notion 7.5).

7.3 Impossibility: Balance of false negative rates unachievable

In this section, we study a popular group fairness notion, equalized odds, in the context of online learning. A natural extension of equalized odds for online settings would require that the false negative rate, i.e. percentage of positive examples predicted incorrectly, is the same across all groups and the same also holds for the false positive rate. We assume that our experts are fair in isolation with respect to both false negative as well as false positive rate. A weaker notion of equalized odds is *equality of opportunity* where the non-discrimination condition is required to be satisfied only for the false negative rate. We first study whether it is possible to achieve the vanishing regret property while guaranteeing α-fairness in composition with respect to false negative rate for arbitrarily small α. When the input is i.i.d., this is trivial as we can learn the best expert in $O(\log d)$ rounds and then follow its advice; since the expert is fair in isolation, this will guarantee vanishing non-discrimination.

In contrast, we show that, in a non-i.i.d. online setting, this goal is unachievable. We demonstrate this in phenomenally benign settings where there are just two groups $\mathcal{G} = \{A, B\}$ that come from a fixed distribution and just two experts that are fair in isolation (with respect to false negative rate) even per round – not only ex post. Our first construction (Theorem 7.1) shows that any no-regret learning algorithm that is group-unaware cannot guarantee fairness in composition, even in instances that are perfectly balanced (each pair of label and group gets 1/4 of the examples) – the only adversarial component is the order in which these examples arrive. This is surprising because such a task is straightforward in the stochastic setting as all hypotheses are non-discriminatory. We then study whether actively using the group identity can correct the aforementioned similarly to how it enables correction against discriminatory predictors [65]. The answer is negative even in this scenario (Theorem 7.2): if the population is sufficiently not balanced, any no-regret learning algorithm will be unfair in composition with respect to false negative rate even if they are not group-unaware.

Group-unaware algorithms. We first present the impossibility result about group-unaware algorithms.

Theorem 7.1. For all $\alpha < 3/8$, there exists $\epsilon > 0$ such that any group-unaware algorithm that satisfies $\mathbb{E}_\sigma\Big[ApxReg_{\epsilon,T}(f)\Big] = o(T)$ for all $f \in \mathcal{F}$ is α-unfair in composition with respect to false negative rate even for perfectly balanced sequences. In particular, for any group-unaware algorithm that ensures vanishing approximate regret[1], there exists an oblivious adversary for assigning labels such that:

- In expectation, half of the population corresponds to each group.
- For each group, in expectation half of its labels are positive and the other half are negative.
- The false negative rates of the two groups differ by α.

Proof sketch. Consider an instance that consists of two groups $\mathcal{G} = \{A, B\}$, two experts $\mathcal{F} = \{h_n, h_u\}$, and two phases: Phase I and Phase II. Group A is the group we end up discriminating against while group B is boosted by the discrimination with respect to false negative rate. At each round t the groups arrive stochastically with probability $1/2$ each, independent of $\sigma^{1:t-1}$.

The experts output a score value in $\hat{\mathcal{Y}} = [0, 1]$, where score $\hat{y}_f^t \in \hat{\mathcal{Y}}$ can be interpreted as the probability that expert f assigns to label being positive in

[1]This requirement is weaker than vanishing regret so the impossibility result applies to vanishing regret algorithms.

round t, i.e. $y(t) = +$. The loss function is the expected probability of error given by $\ell(\hat{y}, y) = \hat{y} \cdot \mathbf{1}\{y = -\} + (1 - \hat{y}) \cdot \mathbf{1}\{y = +\}$. The two experts are very simple: h_n always predicts negative, i.e. $\hat{y}^t_{h_n} = 0$ for all t, and h_u is an unbiased expert who, irrespective of the group or the label, makes an inaccurate prediction with probability $\beta = 1/4 + \sqrt{\epsilon}$, i.e. $\hat{y}^t_{h_u} = \beta \cdot \mathbf{1}\{y(t) = -\} + (1 - \beta) \cdot \mathbf{1}\{y(t) = +\}$ for all t. Both experts are fair in isolation with respect to both false negative and false positive rates: FNR is 100% for h_n and β for h_u regardless the group, and FPR is 0% for h_n and β for h_u, independent of the group. The instance proceeds in two phases:

1. Phase I lasts for $T/2$ rounds. The adversary assigns negative labels on examples with group context B and assigns a label uniformly at random to examples from group A.
2. In Phase II, there are two plausible worlds:
   (a) if the expected probability the algorithm assigns to expert h_u in Phase I is at least $\mathbb{E}_\sigma\left[\sum_{t=1}^{T/2} p^t_{h_u}\right] > \sqrt{\epsilon} \cdot T$ then the adversary assigns negative labels for both groups
   (b) else the adversary assigns positive labels to examples with group context B while examples from group A keep receiving positive and negative labels with probability equal to half.

   We will show that for any algorithm with vanishing approximate regret property, i.e. with $ApxReg_{\epsilon,T}(f) = o(T)$, the condition for the first world is never triggered and hence the above sequence is indeed balanced.

We now point to why the above instance is unfair in composition (the complete proof is provided in Appendix F.1). This stems from the two following claims:

1. In Phase I, any ϵ-approximate regret algorithm needs to select the negative expert h_n most of the times to ensure small approximate regret with respect to h_n. This means that, in Phase I (where we encounter half of the positive examples from group A and none from group B), the false negative rate of the algorithm is close to 1.
2. In Phase II, any ϵ-approximate regret algorithm should quickly catch up to ensure small approximate regret with respect to h_u and hence the false negative rate of the algorithm is closer to β. Since the algorithm is group-unaware, this creates a mismatch between the false negative rate of B (that only receives false negatives in this phase) and A (that has also received many false negatives before).

■

Group-aware algorithms. We now turn our attention to group-aware algorithms, that can use the group context of the example to select the probability of each expert and provide a similar impossibility result. There are three changes compared to the impossibility result we provided for group-unaware algorithms. First, the adversary is not oblivious but instead is adaptive. Second, we do not have perfect balance across populations but instead require that the minority population arrives with probability $b < 0.49$, while the majority population arrives with probability $1 - b$. Third, the labels are not equally distributed across positive and negative for each population but instead positive labels for one group are at least a c percentage of the total examples of the group for a small $c > 0$. Although the upper bounds on b and c are not optimized, our impossibility result cannot extend to $b = c = 1/2$. Understanding whether one can achieve fairness in composition for such values of b and c is an interesting open question.

Theorem 7.2. For any group imbalance $b < 0.49$ and $0 < \alpha < \frac{0.49-0.99b}{1-b}$, there exists $\epsilon_0 > 0$ such that for all $0 < \epsilon < \epsilon_0$ any algorithm that satisfies $\mathbb{E}_\sigma\left[ApxReg_{\epsilon,T}(f)\right] = o(T)$ for all $f \in \mathcal{F}$, is α-unfair in composition.

Proof sketch. The instance has two groups: $\mathcal{G} = \{A, B\}$. Examples with group context A are discriminated against and arrive randomly with probability $b < \frac{1}{2}$ while examples with group context B are boosted by the discrimination and arrive with the remaining probability $1 - b$. There are again two experts $\mathcal{F} = \{h_n, h_p\}$, which output score values in $\hat{\mathcal{Y}} = [0, 1]$, where $\hat{y}_f^t$ can be interpreted as the probability that expert f assigns to label being + in round t. We use the earlier loss function of $\ell(\hat{y}, y) = \hat{y} \cdot \mathbf{1}\{y = -\} + (1 - \hat{y}) \cdot \mathbf{1}\{y = +\}$. The first expert h_n is again pessimistic and always predicts negative, i.e. $\hat{y}_{h_n}^t = 0$, while the other expert h_p is optimistic and always predicts positive, i.e. $\hat{y}_{h_p}^t = 1$. These experts again satisfy fairness in isolation with respect to false negative and false positive rate. Let $c = 1/101^2$ denote the percentage of the input that is about positive examples for A, ensuring that $|S_g^\sigma(FNR)| = \Omega(T)$. The instance proceeds in two phases.

1. Phase I lasts $\Theta \cdot T$ rounds for $\Theta = 101c$. The adversary assigns negative labels on examples with group context B. For examples with group context A, the adversary acts as following:

    - if the algorithm assigns probability on the negative expert below $\gamma(\epsilon) = \frac{99-2\epsilon}{100}$, i.e. $p_{h_n}^t(\sigma^{1:t-1}) < \gamma(\epsilon)$, the adversary assigns negative label.
    - otherwise, the adversary assigns positive labels.

2. In Phase II, there are two plausible worlds:

(a) the adversary assigns negative labels to both groups if the expected number of times that the algorithm selected the negative expert with probability higher than $\gamma(\epsilon)$ on members of group A is less than $c \cdot b \cdot T$, i.e. $\mathbb{E}_\sigma\Big[\mathbf{1}\Big\{t \leq \Theta \cdot T : g(t) = A, p^t_{h_n} \geq \gamma(\epsilon)\Big\}\Big] < c \cdot b \cdot T$.

(b) otherwise she assigns positive labels to examples with group context B and negative labels to examples with group context A.

Note that, as before, the condition for the first world will never be triggered by any no-regret learning algorithm (we elaborate on that below) which ensures that $\mathbb{E}_\sigma |S^\sigma_A(FNR)| \geq c \cdot b \cdot T$.

The proof is based on the following claims, shown in Appendix F.2:

1. In Phase I, any vanishing approximate regret algorithm enters the second world of Phase II.
2. This implies a lower bound on the false negative rate on A, i.e. $FNR(A) \geq \gamma(\epsilon) = \frac{99-2\epsilon}{100}$.
3. In Phase II, any ϵ-approximate regret algorithm assigns large enough probability to the positive expert h_p for group B. This implies an upper bound on the false negative rate on B, i.e. $FNR(B) \leq \frac{1}{2(1-b)}$. Therefore this provides a gap in the false negative rates of at least α.

■

7.4 Main positive result: Balance in accuracy achievable

The negative results of the previous section give rise to a natural question of whether fairness in composition can be achieved for some other fairness metric in an online setting. We answer this question positively by suggesting the *equalized error rates* metric *EER* which captures the average loss over the total number of examples (independent of whether this loss comes from false negative or false positive examples). The relevant subset induced by this metric $\mathcal{S}_g^{\sigma}(EER)$ is the set of all examples coming from group $g \in \mathcal{G}$. We again assume that experts are fair in isolation (Definition 7.1) with respect to equalized error rate and show that a simple scheme where we run separately one instance of multiplicative weights for each group achieves fairness in composition (Theorem 7.3). The result holds for general loss functions (beyond pure classification) and is robust to the experts only being approximately fair in isolation. A crucial property we use is that multiplicative weights not only does not perform worse than the best expert; it also does not perform better [59].

The algorithm. We run separate instances of multiplicative weights with a fixed learning rate η, one for each group. More formally, for each pair of expert $f \in \mathcal{F}$ and group $g \in \mathcal{G}$, we initialize weights $w_{f,g}^1 = 1$. At round $t = 1, 2, \ldots, T$, an example with group context $g(t)$ arrives and the learner selects a probability distribution based to the corresponding weights: $p_f^t = \frac{w_{f,g(t)}^t}{\sum_{j \in \mathcal{F}} w_{j,g(t)}^t}$. Then the weights corresponding to group $g(t)$ are updated exponentially: $w_{f,g}^{t+1} = w_{f,g}^t \cdot (1-\eta)^{\ell_f^t \cdot \mathbf{1}\{g(t)=g\}}$.

Theorem 7.3. For any $\alpha > 0$ and any $\epsilon < \alpha$ such that running separate instances of multiplicative weights for each group with learning rate $\eta = \min(\epsilon, \alpha/6)$ guarantees α-fairness in composition and ϵ-approximate regret of at most $O(|\mathcal{G}| \log(d)/\epsilon)$.

Proof. The proof is based on the property that multiplicative weights performs not only no worse than the best expert in hindsight but also no better. Therefore the average performance of multiplicative weights at each group is approximately equal to the average performance of the best expert in that group. Since the experts are fair in isolation, the average performance of the best expert in all groups is the same which guarantees the equalized error rates desideratum. We make these arguments formal below.

We follow the classical potential function analysis of multiplicative weights but apply bidirectional bounds to also lower bound the performance of the algorithm by the performance of the comparator. For each group $g \in \mathcal{G}$ and every expert $f \in \mathcal{F}$, let $L_{f,g} = \sum_{t:g(t)=g} \ell_f^t \cdot \mathbf{1}\{g(t) = g\}$ be the cumulative loss of expert f in examples with group context g, and $\hat{L}_g = \sum_{t=1}^{T} \sum_{f\in\mathcal{F}} p_f^t \ell_f^t \cdot \mathbf{1}\{g(t) = g\}$ to denote the expected loss of the algorithm on these examples. We also denote the best in hindsight expert on these examples by $f^\star(g) = \arg\min_{f\in\mathcal{F}} L_{f,g}$. Recall that $w_{f,g}^t = (1-\eta)^{\sum_{\tau\leq t:g(\tau)=g} \ell_f^\tau}$ is the weight of expert f in the instance of group g and let $W_{t,g} = \sum_{j\in\mathcal{F}} w_{j,g}^t$ be its potential function.

To show that the algorithm does not perform much worse than any expert, we follow the classical potential function analysis and, since $(1-\eta)^x \leq 1 - \eta x$ for all $x \in [0,1]$ and $\eta \leq 1$, we obtain:

$$W_{t+1,g} = \sum_{j\in\mathcal{F}} w_{j,g}^t (1-\eta)^{\ell_j^t \cdot \mathbf{1}\{g(t)=g\}} \leq \sum_{j\in\mathcal{F}} w_{j,g}^t (1 - \eta \ell_j^t \cdot \mathbf{1}\{g(t) = g\}) = W_{t,g}(1 - \eta \sum_{j\in\mathcal{F}} p_j^t \ell_j^t).$$

By the classical analysis, for all $f \in \mathcal{F}$ and $g \in \mathcal{G}$:

$$w_{f,g}^{T+1} = (1-\eta)^{\sum_{t=1}^{T} \ell_f^t \cdot \mathbf{1}\{g^t=g\}} \leq W_{T+1,g} \leq d \cdot \prod_{t=1}^{T} (1 - \eta \sum_{j\in\mathcal{F}} p_j^t \ell_j^t \cdot \mathbf{1}\{g(t) = g\})$$

where the left inequality follows since all summands of $W_{T+1,g}$ are positive and the right inequality follows by unrolling $W_{T+1,g}$ and using that $W_{1,g} = d$.

Taking logarithms and using that $-\eta - \eta^2 < \ln(1-\eta) < -\eta$ for $\eta < 1/2$, this implies that for all $f \in \mathcal{F}$:

$$\hat{L}_g \leq (1+\eta) \cdot L_{f,g} + \frac{\ln(d)}{\eta} \tag{7.1}$$

We now use the converse side of the inequalities to show that multiplicative weights also does not perform much better than the best expert in hindsight $f^\star(g)$. Using that $(1-\eta)^x \geq 1 - \eta(1+\eta)x$ for all $x \in [0,1]$, we obtain:

$$\begin{aligned} W_{t+1,g} = \sum_{j \in \mathcal{F}} w_{j,g}^t \cdot (1-\eta)^{\ell_j^t \cdot \mathbf{1}\{g(t)=g\}} &\geq \sum_{j \in \mathcal{F}} w_{j,g}^t \cdot \left(1 - \eta(1+\eta) \cdot \ell_j^t \cdot \mathbf{1}\{g(t)=g\}\right) \\ &= W_{t,g} \cdot \left(1 - \eta(1+\eta) \sum_{j \in \mathcal{F}} p_i^t \ell_i^t\right). \end{aligned}$$

Using that $f^\star(g)$ is the best expert in hindsight, we can upper bound $\sum_{j \in \mathcal{F}} w_{j,g}^t \leq d \cdot \max_{j \in \mathcal{F}} w_{j,g}^t = d \cdot \max_{f \in \mathcal{F}} (1-\eta)^{\sum_{t=1}^t \ell_f^t \mathbf{1}\{g^t = g\}}$. Similarly to before, it follows that:

$$d \cdot (1-\eta)^{\sum_{t=1}^T \ell_{f^\star(g)}^t \mathbf{1}\{g^t=g\}} \geq W_{T+1} \geq d \cdot \prod_{t=1}^T \left(1 - \eta(1+\eta) \sum_{j \in \mathcal{F}} p_j^t \ell_j^t\right)$$

which, for $\eta < 1/2$, implies that:

$$\widehat{L_g} \geq (1 - 4\eta) \cdot L_{f^\star(g),g} \tag{7.2}$$

The expected ϵ-approximate regret of this algorithm is at most $6 \cdot |\mathcal{G}|$ times the one of a single multiplicative weights instance (by summing over inequalities (7.1) for all $g \in \mathcal{G}$ and since $\epsilon/6 \leq \eta \leq \epsilon$). What is left to show is that the α-fairness in composition guarantee is satisfied, that is there exists T_0 (function of α and ϵ) such that when the number of examples from each group is at least T_0, the maximum difference between average expected losses across groups is bounded by α. Let $g^\star$ be the group with the smallest average expected loss. We will show that the maximum difference from the average expected loss of any other group g is at most α for $T_0 = \frac{6\ln(d)}{\eta\alpha}$. Since the experts are fair in isolation, we know that

$\frac{L_{f,g}}{|\{t:g^t=g\}|} = \frac{L_{f,g'}}{|\{t:g^t=g'\}|}$ for all $f \in \mathcal{F}$ and $g, g' \in \mathcal{G}$. Combining this with inequalities (7.1) and (7.2) and the fact that the losses are in $[0,1]$ and $\eta \le \alpha/6$, we obtain:

$$\frac{\widehat{L}_g}{|\{t : g(t) = g\}|} - \frac{\widehat{L}_{g^\star}}{|\{t : g(t) = g^\star\}|} \le \frac{(1+\eta)L_{f^\star(g),g}}{|\{t : g(t) = g\}|} + \frac{\ln(d)}{\eta|\{t : g(t) = g\}|} - \frac{(1-4\eta)L_{f^\star(g^\star),g^\star}}{|\{t : g(t) = g^\star\}|}$$
$$\le 5\eta \cdot \frac{L_{f^\star(g^\star),g^\star}}{|\{t : g(t) = g^\star\}|} + \frac{\ln(d)}{\eta \cdot T_0} \le \alpha.$$

■

Remark 7.1. *If the instance is instead only approximately fair in isolation with respect to equalized error rates, i.e. the error rates of the two experts are not exactly equal but within some constant κ, the same analysis implies $(\alpha+\kappa)$-fairness in composition with respect to equalized error rates.*

7.5 Balance in accuracy only when learning is not too good

The reader may be also wondering whether it suffices to just run separate learning algorithms in the two groups or whether multiplicative weights has a special property. In the following theorem, we show that the latter is the case. In particular, multiplicative weights has the property of not doing better than the best expert in hindsight. The main representative of algorithms that do not have such a property are the algorithms that achieve low approximate regret compared to a shifting benchmark (tracking the best expert), which we already discussed in the previous chapter. More formally, approximate regret against a shifting comparator $f^\star = (f^\star(1), \dots, f^\star(T))$ is defined as:

$$ApxReg_{\epsilon,T}(f^\star) = \sum_t p_f^t \ell_f^t - (1+\epsilon)\sum_t \ell_{f^\star(t)}^t.$$

Typical bounds are $\mathbb{E}[ApxReg(f^\star)] = O(\frac{K(f^\star)\cdot\ln(dT)}{\epsilon})$ where $K(f^\star) = 1+\sum_{t=2}^{T} \mathbb{1}\{f^\star(t) \neq f^\star(t-1)\}$ is the number of switches in the comparator. We show that any algorithm that achieves such a guarantee even when $K(f^\star) = 2$ does not satisfy fairness in composition with respect to equalized error rate. This indicates that, for the metric of equalized error rates, the algorithm not being too good is essential.

Theorem 7.4. For any $\alpha < 1/2$ and $\epsilon > 0$, any algorithm that can achieve the vanishing approximate regret property against shifting comparators f of length $K(f) = 2$, running separate instances of the algorithm for each group is α-unfair in composition with respect to equalized error rate.

Proof. Our instance has two groups $\mathcal{G} = \{A, B\}$, two experts $\mathcal{F} = \{f_1, f_2\}$, and three phases described below.

1. Phase I lasts for half of the time horizon $\{1, \dots, T/2\}$ and during this time, we receive examples from group A. At round t, the adversary selects loss $\ell_f^t = 1$ for the expert $f \in \mathcal{F}$ that is predicted with higher probability ($p_f^t \geq 1/2$) and $\ell_h^t = 0$ for the other expert $h \in \mathcal{F} - \{f\}$.
2. Phase II lasts $\sum_{\tau=1}^{T/2} \ell_{f_1}^\tau$ rounds and involves examples in B. The adversary selects losses $\ell_{f_1}^t = 1$ and $\ell_{f_2}^t = 0$.
3. Phase III lasts $\sum_{\tau=1}^{T/2} \ell_{f_2}^\tau$ rounds and again involves examples in B. The adversary now selects losses $\ell_{f_1}^t = 0$ and $\ell_{f_2}^t = 1$.

The instance is fair in isolation with respect to equalized error rates as the cardinality of both groups is the same (half of the population in each group) and the experts make the same number of mistakes in both groups. By construction, the algorithm has expected average loss at least $\frac{1}{2}$ in members of group A.

We now focus on group B. By the shifting approximate regret guarantee and given that there exists a sequence of experts of length 2 that has 0 loss, it holds that the total loss of the algorithm needs to be sublinear on T and, in particular, at most $(\frac{1}{2} - \alpha) \cdot \frac{T}{2}$, which implies an expected error rate of $\frac{1}{2} - \alpha$. Subtracting the two error rates concludes the proof. ■

7.6 Remarks

More information about the paper. The results presented in this chapter are joint work with Avrim Blum, Suriya Gunasekar, and Nathan Srebro [28]. With respect to the equalized error rates, we also show that group-unaware algorithms also suffer from impossibility results. Our work opens up a number of interesting questions with respect to whether other fairness metrics are compatible with the no-regret property. Additionally, in the impossibility result for group-aware algorithms, we heavily used that the adversary is adaptive and there was some imbalance between the two populations; understanding what happens when this is not the case would be interesting.

On balance across groups as a fairness notion. Our work points to an issue that balance notions suffer from. If it is difficult to classify correctly a particular group, balance notions require the decision-maker to jeopardize the performance in other (possibly easily classifiable) groups. Providing bad treatment despite enough confidence about the best alternative is arguably immoral and, in cases such as clinical trials, explicitly illegal. Tackling this concern, in an ongoing joint work with Avrim Blum, we suggest a group fairness notion for online decision-making that, instead of focusing on equality, aims for accuracy in all (possibly overlapping) populations and discuss the arising incentive issues.

On fairness in online decision-making. Dealing with fairness issues in online decision-making has gained much attention over the last few years. One line of work extends individual notions of fairness which require that *similar individuals (with respect to some similarity metric) should be treated similarly* [49] to online settings [90, 57, 62]. Another line of work aims to achieve the so called *meritocratic fairness* [71, 73], which says that an individual/group of higher intrinsic quality should never be selected with smaller probability than less qualified candidates. Regarding notions targeting discrimination against particular groups, beyond our work, there have been nice attempts to tackle important considerations of sequential decision-making. In particular, a line of work points to counterintuitive externalities of using contextual bandit algorithms agnostic to the group identity and suggest that heterogeneity in data can replace the need for exploration [20, 74, 109]. Other works have focused on designing bandit algorithms that restrict the probabilities of selecting a particular group to avoid overexposure or equivalently underexposure [38], or are only given one-sided feedback [21].

One important distinction compared to these works is that we do not assume that the input is i.i.d. over time. A main complication in most of the above works is that the algorithm needs to be very pessimistic throughout exploration to learn the *best fair policy* but subsequently the algorithm can just use this policy over time. In non-i.i.d. settings, the fairness consideration does not only affect an initial stage; the algorithm needs to balance the optimization goal with the fairness constraint throughout all time. Focusing on the simplest extension of adversarial online learning with fairness concerns (all experts assumed to be individually fair), our work sheds light on which notions of fairness are amenable to non-i.i.d. inputs arriving online.

APPENDIX A

SUPPLEMENTARY MATERIAL FROM CHAPTER 2.

A.1 Concentration inequality

Lemma 2.2 (restated). Let $x_1, x_2, \ldots, x_T$ be a sequence of nonnegative random variables, each with $x_t \in [0,1]$, and let $m_t = \mathbb{E}_{t-1}[x_t] = \mathbb{E}[x_t | x_1, \ldots, x_{t-1}]$, the random variable that is the expectation of x_t conditioned on the sequence $x_1, x_2, \ldots, x_{t-1}$. Let $\epsilon > 0$, and $X = \sum_{t=1}^{T} x_t$ and $M = \sum_{t=1}^{T} m_t$. Then, with probability at least $1 - \delta$

$$X - (1+\epsilon)M \leq \frac{(1+\epsilon)\ln(1/\delta)}{\epsilon}$$

and also with probability at least $1 - \delta$

$$(1-\epsilon)M - X \leq \frac{(1+\epsilon)\ln(1/\delta)}{\epsilon}$$

Proof. The proof follows the outline of classical Chernoff bounds for independent variables combined with the law of total expectation to handle the dependence.

First claim. For parameters $b, \lambda > 0$ to be set later, it holds:

$$\mathbb{P}[X - (1+\epsilon)M > b] \leq e^{-\lambda b}\, \mathbb{E}\Big[e^{\lambda(X-(1+\epsilon)M)}\Big] = e^{-\lambda b}\, \mathbb{E}\left[\prod_{t=1}^{T} e^{\lambda(x_t-(1+\epsilon)m_t)}\right] \tag{A.1}$$

We will prove by induction on T that the expectation above is at most 1 if we use $\lambda = \ln(1+\epsilon)$. Given this fact, we can set b such that $e^{-\lambda b} = e^{-\ln(1+\epsilon)b} = \delta$. Using that $\ln(1+\epsilon) \geq \epsilon/(1+\epsilon)$ for all $\epsilon \geq 0$, it follows that $b = \frac{\ln(1/\delta)}{\ln(1+\epsilon)} \leq \frac{(1+\epsilon)\cdot\ln(1/\delta)}{\epsilon}$.

Base of induction for first claim. Now consider the expectation $\mathbb{E}\Big[\prod_{t=1^T} e^{\lambda(x_t-(1+\epsilon)m_t)}\Big]$ for $\lambda = \ln(1+\epsilon)$, we prove by induction on T that this expectation is at most 1.

For the base case of $T = 1$ we have a single random variable $x_1 \in [0,1]$ and its expectation $m_1 = \mathbb{E}[x_1]$. The expectation is $\mathbb{E}\Big[e^{\lambda(x_1-(1+\epsilon)m)}\Big] = \mathbb{E}\Big[e^{\lambda x_1}\Big] \cdot e^{-\lambda(1+\epsilon)m_1}$.

Note that for any value of $x \in [0,1]$, the following simple inequality holds:

$$e^{\lambda x} \leq xe^{\lambda} - x + 1$$

This is true as it holds with equality for $x = 0$ and 1, and the difference is a concave function (as the second derivative of $g(x) = e^{\lambda x} - xe^{\lambda} + x - 1$ is $g''(x) = \lambda^2 e^{\lambda x} \geq 0$), so the inequality is true between the two points. Now write its expectation as:

$$\mathbb{E}\Big[e^{\lambda x_1}\Big] \leq \mathbb{E}\Big[xe^{\lambda} - x_1 + 1\Big] = \mathbb{E}\Big[x \cdot \Big(e^{\lambda} - 1\Big) + 1\Big] = m \cdot \Big(e^{\lambda} - 1\Big) + 1 \leq e^{m\cdot\left(e^{\lambda}-1\right)}.$$

Using this in the expectation of (A.1), we obtain:

$$\mathbb{E}\Big[e^{\lambda(x_1-(1+\epsilon)m)}\Big] \leq e^{m\cdot\left(e^{\lambda}-1\right)} \cdot e^{-\lambda(1+\epsilon)m} = e^{m\left(e^{\lambda}-1-\lambda(1+\epsilon)\right)} \leq 1$$

where the last inequality follows from the choice of $\lambda = \ln(1+\epsilon)$, as the multiplier of m in the exponent with this choice of λ is

$$e^{\lambda} - 1 - \lambda(1+\epsilon) = \epsilon - (1+\epsilon)\ln(1+\epsilon) \leq \epsilon - (1+\epsilon)\Big(\epsilon - \epsilon^2/2\Big) = -\frac{\epsilon^2(1-\epsilon)}{2} < 0.$$

Inductive step for first claim. Now we are ready to prove the general case. Using the law of total expectation, we obtain:

$$\begin{aligned}\mathbb{E}\left[\prod_{t=1}^{T} e^{\lambda(x_t-(1+\epsilon)m_t)}\right] &= \mathbb{E}\left[\prod_{t=1}^{T-1} e^{\lambda(x_t-(1+\epsilon)m_t)} \cdot e^{\lambda(x_T-(1+\epsilon)m_T)}\right] \\ &= \mathbb{E}\left[\prod_{t=1}^{T-1} e^{\lambda(x_t-(1+\epsilon)m_t)} \cdot \mathop{\mathbb{E}}_{T-1}\Big[e^{\lambda(x_T-(1+\epsilon)m_T)}\Big]\right]\end{aligned}$$

where $\mathbb{E}_{t-1}[\cdot]$ is the random variable taking expectation over the last term conditioned on all the previous terms $x_1, \ldots, x_{T-1}$. Note that conditioned on the previous terms, the conditional expectation $\mathbb{E}_{T-1}\left[e^{\lambda(x_T-(1+\epsilon)m_T)}\right]$ is exactly the base case, and hence at most 1 by the above, so we can conclude that

$$\mathbb{E}\left[\prod_{t=1}^{T} e^{\lambda(x_t-(1+\epsilon)m_t)}\right] \leq \mathbb{E}\left[\prod_{t=1}^{T-1} e^{\lambda(x_t-(1+\epsilon)m_t)}\right]$$

and the statement follows by the induction hypothesis.

Second claim. To prove the lower bound, we proceed in an analogous way. For $\lambda = -\ln(1-\epsilon)$, using that $1/(1-\epsilon) \geq 1+\epsilon$, we obtain the equivalent of the inequality (A.1) with $b = \frac{\ln(1/\delta)}{\ln(1/(1-\epsilon))} \leq \frac{\ln(1/\delta)}{\ln(1+\epsilon)}$.

$$\mathbb{P}[(1-\epsilon)M - X > b] \leq e^{-\lambda b}\,\mathbb{E}\left[e^{\lambda((1-\epsilon)M-X)}\right] = e^{-\lambda b}\,\mathbb{E}\left[\prod_{t=1}^{T} e^{\lambda((1-\epsilon)m_t-x_t)}\right] \qquad \text{(A.2)}$$

Regarding a bound on the expectation, consider first a single variable $m_1 = \mathbb{E}[x_1]$.

$$\mathbb{E}\left[e^{-\lambda x_1}\right] \leq \mathbb{E}\left[x_1 e^{-\lambda} - x_1 + 1\right] = m_1\left(e^{-\lambda}-1\right) + 1 \leq e^{m_1\left(e^{-\lambda}-1\right)}$$

We now bound the expectation as

$$\mathbb{E}\left[e^{\lambda((1-\epsilon)m_1-x_1)}\right] \leq e^{\lambda(1-\epsilon)m_1}\,\mathbb{E}\left[e^{-\lambda x_1}\right] \leq e^{\lambda(1-\epsilon)m_1}\cdot e^{m_1\cdot\left(e^{-\lambda}-1\right)} = e^{m_1(\lambda(1-\epsilon)+(e^{-\lambda}-1))} \leq 1$$

where the last inequality follows from the choice of $\lambda = -\ln(1-\epsilon)$, as the multiplier of m in the exponent with this choice of λ is

$$\lambda(1-\epsilon) + (e^{-\lambda}-1) = -(1-\epsilon)\ln(1-\epsilon) - \epsilon \leq (1-\epsilon)\epsilon - \epsilon = -\epsilon^2 < 0.$$

using the fact that $\ln(1-\epsilon) \leq -\epsilon$. The induction then follows as before. ■

A.2 Transforming approximate regret to small-loss guarantees

Lemma 2.3 (restated). Suppose we have a randomized algorithm that takes as input any $\epsilon > 0$ and guarantees that, for some $q \geq 1$ and some function $\Psi(\cdot)$, and any $\delta > 0$, with probability $1 - \delta$, for any time horizon s and any comparator f:

$$(1-\epsilon)\sum_{t=1}^{s} \ell_{A(t)}^{t} \leq \sum_{t=1}^{s} \ell_f^t + \frac{\Psi(\delta)}{\epsilon^q}.$$

Assume using this algorithm over multiple phases (by restarting the algorithm when a phase end). We run each phase τ with $\epsilon_\tau = 2^{-\tau}$ until $\epsilon_\tau \widehat{L}_\tau > \frac{\Psi(\delta)}{(\epsilon_\tau)^q}$ where $\widehat{L}_\tau$ denotes the cumulative loss of the algorithm for phase τ. For any $\delta > 0$, the regret for this multi-phase algorithm is bounded, with probability at least $1 - \delta$ by:

$$\textit{Regret} \quad \leq O\Big((L^\star)^{\frac{q}{q+1}} \Psi\Big(\tfrac{\delta}{\log(L^\star+1)+1}\Big)^{\frac{1}{q+1}} + \Psi\Big(\tfrac{\delta}{\log(L^\star+1)+1}\Big) + 1\Big)$$

Proof. We denote the loss of the algorithm within phase τ as $\widehat{L}_\tau$ and the loss of the best arm within the phase as $L_\tau^\star$. Note that on any phase τ, by our premise about approximate regret on each phase, with probability at least $1 - \delta'$,

$$\widehat{L}^\tau - L_\tau^\star \leq \epsilon_\tau \widehat{L}_\tau + \frac{\Psi(\delta')}{(\epsilon_\tau)^q}$$

The term $\epsilon_\tau \widehat{L}_\tau$ of the right hand side can be split in two terms, i) all but the last round of the phase and ii) the last round. The first term is bounded by $\frac{\Psi(\delta')}{(\epsilon_\tau)^q}$ due to the doubling condition. The second term can be upper bounded by ϵ_τ since the losses are in $[0, 1]$. Hence, for phase τ, with probability $1 - \delta'$:

$$\widehat{L}_\tau - L_\tau^\star \leq \frac{2 \cdot \Psi(\delta')}{(\epsilon_\tau)^q} + \epsilon_\tau.$$

Letting Γ denote the last phase and summing over the phases, we have:

$$\begin{aligned}
\widehat{L} - L^\star &\le \sum_{\tau=0}^{\Gamma-1} \frac{2\Psi(\delta')}{(\epsilon_\tau)^q} + \sum_{\tau=0}^{\Gamma-1} \epsilon_\tau + \epsilon_\Gamma \widehat{L}_\Gamma + \frac{\Psi(\delta')}{(\epsilon_\Gamma)^q} \\
&\le 2\Psi(\delta') \sum_{\tau=0}^{\Gamma} \frac{1}{2^{-q\tau}} + \sum_{\tau=0}^{\Gamma-1} 2^{-\tau} + \epsilon_\Gamma \widehat{L}_\Gamma \\
&\le 2\Psi(\delta') \cdot \frac{2^{q(\Gamma+1)} - 1}{2^q - 1} + 2 + \epsilon_\Gamma \widehat{L}_\Gamma \\
&\le 4\Psi(\delta') \frac{1}{(\epsilon_\Gamma)^q} + 2 + \epsilon_\Gamma \widehat{L}_\Gamma && \text{Since } q \ge 1 \\
&\le 4\left(\frac{\Psi(\delta')}{(\epsilon_\Gamma)^q}\right)^{1/(q+1)} \cdot \left(2^q \frac{\Psi(\delta')}{(\epsilon_{\Gamma-1})^q}\right)^{q/(q+1)} + \left(\epsilon_\Gamma \widehat{L}_\Gamma\right)^{1/(q+1)} \cdot \left(\epsilon_\Gamma \widehat{L}_\Gamma\right)^{q/(q+1)} + 2 \\
&\le 2^{q+2}\left(\frac{\Psi(\delta')}{(\epsilon_\Gamma)^q}\right)^{1/(q+1)} \cdot \left(\epsilon_P \widehat{L}_{\Gamma-1}\right)^{q/(q+1)} + \left(\frac{\Psi(\delta')}{(\epsilon_\Gamma)^q}\right)^{1/(q+1)} \cdot \left(\epsilon_\Gamma \widehat{L}_\Gamma\right)^{q/(q+1)} + 2
\end{aligned}$$

Thus we conclude that:

$$\widehat{L} - L^\star \le O\left((\Psi(\delta'))^{1/(q+1)} \cdot \left(\widehat{L}\right)^{q/(q+1)} + 1 \right)$$

To replace the dependence of $\widehat{L}$ by $L^\star$, we apply Young's inequality, the approximate regret property, and the sub-additivity property. For simplicity of presentation, we remove the multiplicative and additive constants and use $a = q/(q+1)$ so that the analysis is clear for different small-loss powers.

$$\widehat{L} - L^\star \le \Psi(\delta')^{1-a} \cdot \left(\widehat{L}\right)^a \le (1-a)\Psi(\delta') + a\widehat{L} \Rightarrow$$

$$\widehat{L} \le \frac{1}{1-a} L^\star + \Psi(\delta')$$

Replacing to the previous guarantee and applying the subadditivity property

$$\begin{aligned}
\widehat{L} - L^\star \le \Psi(\delta')^{1-a} \cdot \left(\widehat{L}\right)^a &\le \Psi(\delta')^{1-a} \cdot \left(\frac{1}{1-a} L^\star + \Psi(\delta')\right)^a \\
&\le \frac{1}{1-a} (L^\star)^a \Psi(\delta')^{1-a} + \Psi(\delta')
\end{aligned}$$

Since there are at most $\log(L^\star + 1) + 1$ phases, setting $\delta' = \frac{\delta}{\log(L^\star+1)+1}$ suffices for the high probability statements to hold for all phases. ■

APPENDIX B
SUPPLEMENTARY MATERIAL FROM CHAPTER 3.

B.1 Active arm elimination with enlarged confidence intervals

Lemma 3.1 (restated). Assume that c is a valid upper bound for the total corruption and we run Active Arm Elimination with $\mathsf{wd}(a,t) = \sqrt{\frac{\log(2kT/\delta)}{n(a,t)}} + \frac{c}{n(a,t)}$. Then, with probability at least $1-\delta$, arm $a^\star$ never becomes eliminated.

Proof. The crux of the proof lies in establishing that, with high probability, the upper bound of the confidence interval of $a^\star$ never becomes lower than the lower bound of the confidence interval of any other arm a and therefore $a^\star$ does not become eliminated. More formally, let $\tilde{\theta}(a,t)$ be the empirical mean of the rewards from arm a until time t (that is the uncorrupted part) and $\tilde{\mu}(a,t)$ be the empirical mean of the corrupted rewards from the same arm. We also denote by $n(a,t)$ the number of times we selected arm a till then. Recall that $\mu(a)$ is the mean of arm a. By Hoeffding inequality, for any arm a, with probability at least $1-\delta'$:

$$|\tilde{\theta}(a,t) - \mu(a)| \leq \sqrt{\frac{\log(2/\delta')}{n(a,t)}}.$$

We set $\delta' = \delta/kT$ to establish that this holds for all arms and all time steps (after arm a has been played $n(a)$ times). As a result, for any arm a and any time t:

$$\tilde{\theta}(a,t) \leq \mu(a) + \sqrt{\frac{\log(2kT/\delta)}{n(a,t)}} \quad \text{and} \quad \tilde{\theta}(a^\star,t) \geq \mu(a^\star) - \sqrt{\frac{\log(2kT/\delta)}{n(a^\star,t)}}.$$

Let's focus now on the actual (corrupted) empirical means. Since c is a valid upper bound on the total corruption then the (corrupted) empirical means can be affected by at most absolute corruption c. Hence:

$$\tilde{\mu}(a,t) \leq \tilde{\theta}(a,t) + \tfrac{c}{n(a,t)} \quad \text{and} \quad \tilde{\mu}(a^\star,t) \geq \tilde{\theta}(a^\star,t) - \tfrac{c}{n(a^\star,t)}.$$

Combining the above inequalities with the fact that the actual mean of $a^\star$ is higher than the one of a, i.e. $\mu(a^\star) \geq \mu(a)$, we establish that $\tilde{\mu}(a,t) - \tilde{\mu}(a^\star,t) \leq \mathsf{wd}(a,t) + \mathsf{wd}(a^\star,t)$ and therefore arm $a^\star$ is not eliminated. Since this holds for all times and arms, the lemma follows. ■

Lemma 3.2 (restated). Assume that c is a valid upper bound for the total corruption and we run Active Arm Elimination with $\mathsf{wd}(a,t) = \sqrt{\frac{\log(2kT\delta)}{n(a,t)}} + \frac{c}{n(a,t)}$. Then, with probability at least $1-\delta$, all arms $a \neq a^\star$ become eliminated after $N(a) = \frac{36\log(2kT/\delta)+6c}{\Delta(a)^2}$ plays.

Proof. The proof stems from the following observations. By Lemma 3.1, arm $a^\star$ is with high probability never eliminated. Assume that arm a is played $N(a)$ times and let $\tau(a)$ be the time that arm a is played for the $N(a)$-th time. We will show that, with high probability, arm a is dominated by $a^\star$ at this point, i.e. after $N(a)$ plays of arm a, with high probability, the lower confidence bound of arm $a^\star$ is above the upper confidence bound of arm a.

More formally, let again $\tilde{\theta}(a,t)$ be the empirical mean of the rewards from arm a until time t (that is the uncorrupted part) and $\tilde{\mu}(a)$ be the empirical mean of the corrupted rewards from the same arm. By Hoeffding inequality, for any arm a, with probability at least $1-\delta'$:

$$|\tilde{\theta}(a,t) - \mu(a)| \leq \sqrt{\frac{\log(2/\delta')}{n(a,t)}}. \tag{B.1}$$

As a result, setting again $\delta' = \delta/kT$, after $N(a) = \frac{36\log(2kT/\delta)+6c}{\Delta(a)^2}$ plays of arm a, the empirical uncorrupted mean of a is, with high probability, at most:

$$\tilde{\theta}(a, \tau(a)) \leq \mu(a) + \sqrt{\tfrac{\log(2KT/\delta)}{N(a)}} \leq \mu(a) + \tfrac{\Delta(a)}{6}.$$

Similarly, the empirical stochastic mean of $a^\star$ is, with high probability, at least:

$$\tilde{\theta}(a^\star, \tau(a)) \geq \mu(a^\star) - \sqrt{\tfrac{\log(2kT/\delta)}{N(a)}} \geq \mu(a^\star) - \tfrac{\Delta(a)}{6}$$

Since the corruptions are upper bounded by $C \leq c$, they can only contribute to a decrease in the average empirical (corrupted) means by at most $\frac{\Delta(a)}{6}$ which is not enough to circumvent the gap $\Delta(a)$ due to the choice of $N(a)$:

$$\tilde{\mu}(a^\star, \tau(a)) \geq \tilde{\theta}(a^\star, \tau(a)) - \tfrac{c}{N(a)} \geq \tilde{\theta}(a^\star, \tau(a)) - \tfrac{\Delta(a)}{6}$$

$$\tilde{\mu}((a, \tau(a)) \leq \tilde{\theta}(a, \tau(a)) + \tfrac{c}{N(a)} \leq \tilde{\theta}(a, \tau(a)) + \tfrac{\Delta(a)}{6}$$

Finally, at time $\tau(a)$, the width of both arms, played $N(a)$ times at this point, is:

$$\mathsf{wd}(a^\star, \tau(a)) = \mathsf{wd}(a, \tau(a)) = \sqrt{\tfrac{\log(2kT/\delta)}{N(a)}} + \tfrac{c}{N(a)} \leq \tfrac{\Delta(a)}{3}.$$

Combining the above, with probability $1 - \delta$, if arm a is not eliminated until then, it is eliminated at time $\tau(a)$, i.e. $\tilde{\mu}(a^\star, \tau(a)) - \tilde{\mu}(a, \tau(a)) \geq \mathsf{wd}(a^\star, \tau(a)) + \mathsf{wd}(a, \tau(a))$:

$$\begin{aligned}
\tilde{\mu}(a^\star, \tau(a)) - \tilde{\mu}(a, \tau(a)) &\geq \tilde{\theta}(a^\star, \tau(a)) - \tilde{\theta}(a, \tau(a)) - 2 \cdot \frac{\Delta(a)}{6} \\
&\geq \mu(a^\star) - \mu(a) - 4 \cdot \frac{\Delta(a)}{6} \\
&\geq \mu(a^\star) - \mu(a) - 4 \cdot \frac{\Delta(a)}{6} - 2 \cdot \frac{\Delta(a)}{6} + \mathsf{wd}(a^\star, \tau(a)) + \mathsf{wd}(a, \tau(a)) \\
&\geq \mu(a^\star) - \mu(a) - \Delta(a) + \mathsf{wd}(a^\star, \tau(a)) + \mathsf{wd}(a, \tau(a)) \\
&\geq + \mathsf{wd}(a^\star, \tau(a)) + \mathsf{wd}(a, \tau(a))
\end{aligned}$$

■

B.2 Fast-slow active arm elimination race intervals

In this section, we provide the proof of Theorem 3.1. To handle the corruption, we bound with high probability the total corruption experienced by the slow active arm elimination instance S (Lemma 3.3). To deal with an adaptive adversary, we need a martingale concentration inequality; specifically we apply a Bernstein-style inequality introduced in [23] (Lemma B.1).

Lemma B.1 (Lemma 1 in [23]). Let $X_1, \ldots, X_T$ be a sequence of real-valued random numbers. Assume, for all t, that $X_t \leq R$ and that $\mathbb{E}[X_t | X_1, \ldots, X_{t-1}] = 0$ for $R > 0$. Also let $V = \sum_{t=1}^{T} \mathbb{E}[X_t^2 | X_1, \ldots, X_{t-1}]$. Then, for any $\delta > 0$:

$$\mathbb{P}\left[\sum_{t=1}^{T} X_t > R \ln(1/\delta) + \frac{e-2}{R} \cdot V\right] \leq \delta$$

Lemma 3.3 (restated). If the total corruption is $C \leq c$ then the slow active arm elimination algorithm S observes, with probability at least $1-\delta$, corruption of at most $\ln(1/\delta) + 3$ during its exploration phase (when picked with probability $1/c$).

Proof. The first observation is that the expected corruption encountered by algorithm S is at most a constant (total corruption of C encountered with probability $1/c$). The rest of the proof focuses on bounding the variance of this random variable (actual corruption encountered by the layer). Crucially, since we want to allow the adversary to be adaptive, we should not assume independence across rounds but only conditional independence (conditioned on the history) and this is why some more involved concentration inequality is necessary. Therefore we create a martingale sequence (actual corruption minus expected corruption) and apply a Bernstein-style concentration inequality.

Let Z_a^t be the corruption that is observed by the exploration phase of the algorithm if arm a is selected. For every round t, if adversary selects corruption C_a^t then Z_a^t is therefore a random variable equal to C_a^t with probability $1/c$ and 0 otherwise. Given that the adversary is adaptive and may select the corruptions based on the realizations of the previous rounds, we need to use an appropriate concentration inequality. We use a Bernstein-style inequality, introduced in [23] (Lemma B.1). Initially we resolve the randomness conditioning on $\ell = \mathsf{S}$ (the slow algorithm is selected). Since active arm elimination is deterministic, conditioned on selecting algorithm S, the selected arm is deterministic. Let $a(\mathsf{S}, t)$ be the arm that would be selected if $\ell = \mathsf{S}$ (which happens with probability $1/c$). The martingale sequence is now

$$X_t = Z^t_{a(\mathsf{S},t)} - \mathbb{E}\Big[Z^t_{a(\mathsf{S},t)} \mid \mathcal{H}(1 : t-1)\Big],$$

where $\mathcal{H}(1 : t)$ corresponds to the history up to round t. Note that

$$\begin{aligned}\mathbb{E}\Big[X_t^2|X_1,\dots,X_{t-1}\Big] &= \frac{1}{c}\bigg(C_{a(\mathsf{S},t)} - \frac{C_{a(\mathsf{S},t)}}{c}\bigg)^2 + \frac{c-1}{c}\bigg(\frac{C_{a(\mathsf{S},t)}}{c}\bigg)^2 \\ &= \frac{(C_{a(\mathsf{S},t)})^2}{c}\bigg(\frac{c-1}{c}\bigg)^2 + \frac{c-1}{c}\bigg(\frac{C_{a(\mathsf{S},t)}}{c}\bigg)^2 \leq 2\cdot\frac{C_{a(\mathsf{S},t)}}{c}.\end{aligned}$$

The last inequality holds as $C^t_{a(\mathsf{S},t)} \in [0,1]$ and $C_{a(\mathsf{S},t)} \leq c$.

Therefore, summing over all the rounds,

$$V = \sum_t \mathbb{E}\Big[X_t^2|X_1,\dots,X_{t-1}\Big] \leq \sum_t 2\frac{C_{a(\mathsf{S},t)}}{C} \leq \frac{2}{C}\cdot\Bigg(\sum_t \max_a C_a^t\Bigg) \leq 2.$$

A trivial upper bound of $|X_t|$ is $R = 1$, since the rewards are in $[0,1]$. Applying Lemma B.1, we show that, w.p. $1-\delta$:

$$\sum_t X_t \leq \ln(1/\delta) + 2(e-2) \leq \ln(1/\delta) + 2$$

The bound of the statement then for the corruption experienced by S then follows by adding the expected corruption $\mathbb{E}[\sum_t Z_{a(\mathsf{S},t)} \mid \mathcal{H}(1:t-1)] \leq 1$:

$$\sum_t Z^t_{a(\mathsf{S},t)} = \sum_t X_t + \mathbb{E}\left[\sum_t Z_{a(\mathsf{S},t)} \mid \mathcal{H}(1:t-1)\right] \leq \ln(1/\delta) + 3.$$

■

Theorem 3.1 (restated) With probability $1-\delta$, the fast-slow active arm elimination has regret $O\Big(\sum_{a\neq a^\star} \frac{\log(kT/\delta)}{\Delta(a)}\Big)$ for the stochastic case and $O\Big(k \cdot c \cdot \sum_{a\neq a^\star} \frac{(\log(kT/\delta))^2}{\Delta(a)}\Big)$ for the C-corrupted case with $C \leq c$.

Proof. The fast and slow algorithms are run with widths (for $\ell \in \{\mathsf{F}, \mathsf{S}\}$):

$$\mathsf{wd}^\ell(a) = \sqrt{\frac{\log(8kT/\delta)}{n^\ell(a)}} + \frac{c^\ell}{n^\ell(a)}$$

where $c^{\mathsf{F}} = 0$ for the fast instance and $c^{\mathsf{S}} = \log(8kT/\delta) + 3$ for the slow instance S.

Stochastic case. Let $\delta^{st}_\ell = \delta/4$ be the failure probability of instance $\ell \in \{\mathsf{F}, \mathsf{S}\}$ when the input is stochastic. If we are in the stochastic case, the total corruption is 0 and hence less than both c^{F} and c^{S}. By Lemma 3.2 with probability $1-\delta^{st}_\ell$ for all arms a, it holds that arm $a \neq a^\star$ is eliminated after at most $N(a)$ plays where:

$$N(a) = \frac{36\log(2kT/\delta^{st}_\ell) + 6\log(2kT/\delta^{st}_\ell)}{\Delta(a)^2} = \frac{42\log(2kT/\delta^{st}_\ell)}{\Delta(a)^2}. \tag{B.2}$$

For a pseudoregret guarantee, the result follows directly by multiplying the number of plays for any suboptimal arm $a \neq a^\star$ with its expected contribution to regret every time it is selected which is $\Delta(a)$. Setting $\delta = 1/T$ makes the contribution of the failure probability to regret at most a constant since $\delta \cdot T = 1$.

For a high-probability guarantee, the regret coming from a suboptimal arm a after $N(a)$ plays may be more than the expected regret $N(a)\Delta(a)$. We define as in Lemma 3.2, $\tilde{\theta}(a,t)$ to be the empirical mean of the rewards from arm a until time t and $n(a,t)$ to be the corresponding number of plays. By the Hoeffding inequality there, i.e. Eq. (B.1), for any arm a, with probability at least $1-\delta_\ell^{st}$ (this corresponds to the same failure probability we have already considered):

$$|\tilde{\theta}(a,t)-\mu(a)| \leq \sqrt{\frac{\log\left(2kT/\delta_\ell^{st}\right)}{n(a,t)}}.$$

Hence, the total reward of $a \neq a^\star$ after $N(a)$ plays is at most:

$$\sqrt{N(a)\cdot\log(2kT/\delta_\ell^{st})} = N(a)\cdot\sqrt{\frac{\log(2kT/\delta_\ell^{st}}{N(a)}} \leq N(a)\cdot\Delta(a).$$

which implies that the guarantee also holds with high probability.

Corrupted case. The most interesting case is the C-corrupted setting for some $C \leq c$. Let $\delta_{\mathsf{S}}^{co} = \delta/4$ be the failure probability in Lemma 3.3. By Lemma 3.3, with probability at least $1-\delta_{\mathsf{S}}^{co}$, the actual corruption experienced by the slow active arm elimination algorithm is at most $\ln(1/\delta_{\mathsf{S}}^{co})+3$. Therefore, similar to the stochastic case, we can obtain a high-probability guarantee on the regret coming from selecting suboptimal arms in the slow active arm elimination instance S.

What is left is to bound the regret coming from the fast active arm elimination instance F. Towards this goal, we bound the number of times that a suboptimal arm is played in the fast instance by the expected time that it remains active at the slow instance. By Eq. B.2, with probability at least $1-\delta_{\mathsf{S}}^{st}-\delta_{\mathsf{S}}^{co}$, arm a is played in the slow instance, at most $N_{\mathsf{S}}(a)$ times where:

$$N_{\mathsf{S}}(a) = \frac{42\log(2kT/\delta_{\mathsf{S}}^{st})}{\Delta(a)^2}.$$

For a pseudoregret, we can directly use this bound of the slow instance to bound the number of times a is played in the fast instance. In particular, for any play of arm a in the slow instance, there are in expectation at most $k \cdot c \cdot N_{\mathsf{S}}(a)$ total rounds as every move in the slow instance occurs with probability $1/c$ and at least $1/k$ of these moves are plays of a while it is still active. Hence, arm a is eliminated in the fast instance F after $k \cdot c \cdot N_{\mathsf{S}}(a)$ rounds and, till then, it contributes $\Delta(a)$ to the regret in expectation when played, which provides the pseudoregret guarantee.

For a high probability guarantee, let $\delta_{\mathsf{S}}^{move} = \delta/4kT$ and observe that with probability at least $1 - \delta_{\mathsf{S}}^{move}$, we make one move at the slow instance S at most every $O\big(c \log\big(1/\delta_{\mathsf{S}}^{move}\big)\big)$ moves at the fast instance F. This can be seen by considering the following process: One tosses coins with bias $p = 1/c$ until she observes heads for the first time (heads is the p-biased event). After M tosses of the coins the probability that no heads have arrived is at most $(1-p)^M$. To ensure that this is less than $\delta_{\mathsf{S}}^{move}$, we need to wait $M \geq \frac{\log\left(1/\delta_{\mathsf{S}}^{move}\right)}{\log(\frac{1}{1-p})}$, which is achieved by $M = \frac{\log(1/\delta_{\mathsf{S}}^{move})}{p/(1-p)}$.

By union bound on the failure probabilities for each such draw, we obtain that, with failure probability $\delta_{\mathsf{S}}^{el} = k \cdot N_{\mathsf{S}}(a) \cdot \delta_{\mathsf{S}}^{move} \leq \delta/4$ (since $N_{\mathsf{S}}(a) \leq T$), arm a gets inactivated in F after at most $N_{\mathsf{F}}(a)$ plays where:

$$N_{\mathsf{F}}(a) = k \cdot N_{\mathsf{S}}(a) \cdot c \cdot \log(1/\delta_{\mathsf{S}}^{el}) = \frac{42 \cdot c \cdot k \cdot \left(\log(8KT/\delta)\right)^2}{\Delta(a)^2}.$$

The last part is to prove that the regret experienced throughout those rounds is not too large. This follows again by Eq. (B.1) in a way analogous to the stochastic case. The total failure probability is at most $\delta_{\mathsf{F}}^{st} + \delta_{\mathsf{S}}^{st} + \delta_{\mathsf{S}}^{co} + \delta_{S}^{el} = \delta$. ■

B.3 Multi-layer active arm elimination race

Theorem 3.2 (restated) With probability $1-\delta$, the multi-layer active arm elimination race has regret in the agnostic C-corrupted case bounded by:

$$O\left(\sum_{a\neq a^*} \frac{k\cdot C\cdot \log(kT/\delta)+\log(T)}{\Delta(a)}\cdot \log(kT/\delta)\right).$$

Proof. The proof is similar to Theorem 3.1 with layer $\ell^\star = \arg\min_\ell\left[2^\ell > C\right]$ playing the role of the slow instance S and smaller layers behave as the fast layer F.

Robust layers are essentially stochastic. For layers $\ell \geq \ell^\star$, it holds that the total corruption they experience is, in expectation, less than 1 since $C\cdot 2^{-\ell} \leq 1$ and, as in Lemma 3.3, with high probability less than $c^\ell = \log(4kT\cdot\log T/\delta)+3$. Hence, as in the stochastic case of Theorem 3.1, we establish a $O\left(\frac{\log\left(2KT/\delta_\ell^{st}\right)}{\Delta(a)}\right)$ bound on the regret caused by any suboptimal arm a, with failure probability $\delta_\ell^{st} = \delta/(2\log T)$. Since there are $\log(T)$ such levels, the regret coming from these layers is upper bounded by the second term of the theorem with failure probability $\delta/2$.

Not robust layers eventually corrected by $\ell^\star$. For $\ell < \ell^\star$, we treat them similarly to the fast layer F in the corrupted case of Theorem 3.1. In particular, similar to the proof there, we upper bound the number of times any suboptimal arm is played in layer $\ell^\star$, as in Eq. (B.2). Then we can bound the number of times that this arm is played in faster layers as in the proof there with $c = 2^{-\ell^\star}$. Since, we do not know the corruption C in advance (and it is adaptively selected), we need to take a union bound over all the number of layers as well which results to selecting failure probabilities $\delta_\ell^{co} = \delta_\ell^{el} = \delta/(2\log T)$. Finally, to be agnostic to C, we use $c = 2^{\ell^\star}$ instead of $c = C$; this only increases regret by a constant factor. ■

APPENDIX C
SUPPLEMENTARY MATERIAL FROM CHAPTER 4

In this section, we provide the proof of the lemma connecting spread to absolute and squared loss. Before doing so, we provide a useful auxiliary lemma.

Lemma C.1. For odd $T = 2n + 1$, one pair (A_T, B_T) minimizing either absolute or squared loss subject to the constraints of the spread definition is $A_{2n+1} = (0 \dots 2n)$ and $B_T = (n \dots n)$.

Proof. First we show that there exists a B_T minimizing the loss with $b_i = b_j$ for all i, j. Assume otherwise; then there exist two subsequent i, j with $b'_i > b'_j$. Since $a_i < a_j + 1$ by the assumption on spread, $\min_{x \in b_i, b_j}\{\ell(a_i, b) + \ell(a_j, b)\} \leq \ell(a_i, b_i) + \ell(a_j, b_j)$. Applying this recursively, we conclude that such a B_T exists.

Second, we show that there exist an A_T that consists of elements $a_{i+1} = a_i + 1$. Since the elements of B_T are all equal to b, the sequence $\sum_{i=0}^{2n} \ell(a_i, b)$ is minimized for both absolute and squared loss when $a_i = b + i - n$.

Last, the exact value of b does not make a difference and therefore we can set it to be $b = n$ concluding the lemma. ■

Lemma 4.1 restated: For absolute loss, $\ell_1(A, B) = \sum_i |a_i - b_i|$, the spread of ℓ_1 is $S_{\ell_1}(m) \leq \sqrt{5m}$. For squared loss, $\ell_2(A, B) = \sum(a_i - b_i)^2$, the spread of ℓ_2 is $S_{\ell_2}(m) \leq \sqrt[3]{14m}$.

Proof. It will be easier to restrict ourselves to odd $T = 2n + 1$ and also assume that $T \geq 3$. This will give an upper bound on the spread (which is tight up to small constant factors). By Lemma C.1, a pair of sequence minimizing absolute/squared loss is $A_T = (0, \ldots, 2n)$ and $B_T = (n, \ldots, n)$. We now provide bounds on the spread based on this sequence, that is we find a $T = 2n + 1$ that satisfies the inequality $\ell(A_T, B_T) \leq m$.

Absolute loss: The absolute loss of the above sequence is:

$$\ell(A_T, B_T) = 2 \cdot \sum_{j=1}^{n} j = 2 \cdot \frac{n(n+1)}{2} = n(n+1) = \frac{T-1}{2} \cdot \frac{T+1}{2} = \frac{T^2-1}{4}.$$

A T that makes $\ell(A_T, B_T) \geq m$ is $T = \sqrt{4m+1}$. Therefore, for absolute loss $S_\ell(m) \leq \sqrt{5m}$, since $m \geq 1$

Squared loss: The squared loss of the above sequence is:

$$\ell(A_T, B_T) = 2 \cdot \sum_{j=1}^{n} j^2 = 2 \cdot \frac{n(n+1)(2n+1)}{6} = \frac{(T^2-1) \cdot T}{12} = \frac{T^3 - T}{12} \geq \frac{8T^3}{9 \cdot 12} = \frac{2T^3}{27}$$

where the inequality holds because $T \geq 3$.

A T that makes $\ell(A_T, B_T) \geq m$ is $T = \sqrt[3]{14m}$. Therefore, for squared loss $S_\ell(m) \leq \sqrt[3]{14m}$. ■

APPENDIX D
SUPPLEMENTARY MATERIAL FROM CHAPTER 5.

D.1 Concave reward curves

In this section, we investigate conditions under which throughput, social welfare and revenue satisfy the conditions of theorem 5.6. In particular, we first show that the respective reward curves $R(q) = qI(q)$ are concave. We then prove that the concave reward curves assumption implies the non-increasing (quantiles) per-ride rewards assumption.

Lemma D.1. Revenue (i) satisfies the assumptions of Theorem 5.6 under regular value distributions, Throughput (ii) and Social Welfare (iii) satisfy the assumptions under any value distribution.

Proof. We drop the subscripts throughout this proof to simplify notation. We begin by considering (i) revenue, for which the result holds due to the fact that the reward curve is concave if and only if the distribution is regular (cf. Proposition 3.10 in [66]). For (ii) throughput, $R(q) = q \cdot I(q) = q$ is a linear function of q for any value distribution and thus concave.

Lastly, for (iii) social welfare, we use the so-called hazard rate $h(y) = \frac{f(y)}{1-F(y)}$ of a distribution F with density f. Given F, denote by $p(q)$ and $q(p)$ a price as a function of its corresponding quantile and vice-versa. Then, by the definition of hazard rate:

$$q(p) = \exp\left(- \int_0^{p(q)} h(y)dy \right) \tag{D.1}$$

Taking logarithms and differentiating, we obtain:

$$-\frac{1}{q(p)} = h(p(q))\frac{dp(q)}{dq} \tag{D.2}$$

Hence, as $R(q(p)) = q(p) \cdot I(q(p))$ and $f(p) = (1 - F(p))h(p) = q(p)h(p)$ we have

$$R(q) = \int_{p(q)}^{\infty} vf(v)dv = \int_{p(q)}^{\infty} vh(v)\exp\left(-\int_0^v h(y)dy\right)dv$$

The first derivative $\frac{dR(q)}{dq}$ of $R(q)$ is equal to

$$-p(q)h(p(q))\exp\left(-\int_{y=0}^{p(q)} h(y)dy\right)\frac{dp(q)}{dq} = \frac{p(q)\exp\left(-\int_{y=0}^{p(q)} h(y)dy\right)}{q(p)} = p(q),$$

where the first equality comes from Equation (D.2), the second from (D.1).

The second derivative is then given by

$$\frac{d^2R(q)}{dq^2} = \frac{dp(q)}{dq} = -\frac{1}{qh(p(q))} = -\frac{1-F(p(q))}{f(p(q))q(p)} < 0,$$

which concludes the proof of the Lemma. ■

Lemma D.2. If a function $I(\cdot)$ has the property that $qI(q)$ is concave, then $I(\cdot)$ is non-increasing. In particular, if some objective satisfies the concave reward curves assumption, it also satisfies the non-increasing (in quantiles) per-ride rewards assumption.

Proof. Suppose the statement was not true, then there must exist q_1, q_2 with $0 < q_1 < q_2$ such that $I(q_1) < I(q_2)$. Let $A = \frac{q_1}{q_2}$. Then

$$\begin{aligned} q_1 I(q_2) &= A \cdot q_2 I(q_2) = A \cdot q_2 I(q_2) + (1-A) \cdot 0 \cdot I(0) \\ &\leq (A \cdot q_2 + (1-A) \cdot 0) I(A \cdot q_2 + (1-A) \cdot 0) = q_1 I(q_1), \end{aligned}$$

where the inequality follows from Jensen's inequality since the function $qI(q)$ is a concave function. As $q_1 > 0$, it follows that $I(q_2) \leq I(q_1)$ and we therefore arrive at a contradiction. ■

D.2 Irreducibility of the priced system

We justify here our assumption from Section 5.2 that the infinite-unit solutions we obtain induce a connected graph; to do so, we first need to assume that the graph created by edges (i, j) on which $\phi_{ij} > 0$ is strongly connected, that is, the directed graph with edge-set $\left\{(i, j) : \phi_{ij} > 0\right\}$ contains a path from any node to any other. We then prove that given any solution to the infinite-unit pricing problem, there exists a solution with arbitrarily close objective that also induces a strongly connected graph. For simplicity, we assume that throughput is the objective, yet the extension to other objectives is immediate. Throughout this section we work with the flow $f_{ij,\infty}(\mathbf{q})$ induced by demands in the infinite-unit system, but suppress all dependencies on ∞ in the notation.

Theorem D.1 (Irreducible Markov Chain)**.** Let $\epsilon > 0$. Suppose quantiles $\mathbf{q}$ induce a steady-state rate of units $f_{ij,\infty}(\mathbf{q})$ on k components; then there exist quantiles $\mathbf{q}'$ that induce $f_{ij,\infty}(\mathbf{q}')$ such that the graph with edge-set $\{(i, j) : f_{ij,\infty}(\mathbf{q}') > 0\}$ is strongly connected and the throughput with $\mathbf{q}'$ is at least $(1 - \epsilon)$ times that of $\mathbf{q}$ in the infinite-unit system.

Proof. Notice first that we may assume without loss of generality that $\mathbf{q}$ fulfills the demand circulation property; indeed, whether an arc has non-zero demand on it is independent of the number of units and by Lemma 5.1 there is a solution in the relaxation (with demand circulation) that has an elevated throughput that is no less than $\sum_{i,j} f_{ij}(\mathbf{q})$. To prove the theorem we repeatedly increase demand on some edges (i, j) with $\phi_{ij} > 0$ and $f_{ij}(\mathbf{q}) = 0$, but also decrease demand on some edges $(\bar{i}, \bar{j})$ with $f_{\bar{i}\bar{j}}(\mathbf{q}) > 0$. Equivalently, we increase quantiles q_{ij} and decrease quantiles $q_{\bar{i}\bar{j}}$. To ensure that edges of the second kind do not have their flow reduced by too much relative to $f_{\bar{i}\bar{j}}(\mathbf{q})$, we set

$$\delta = \frac{\epsilon}{k} \times \min\left\{ \min_{i,j}\left\{ f_{ij}(\mathbf{q}) : f_{ij}(\mathbf{q}) > 0 \right\}, \min_{i,j}\left\{ \phi_{ij} - f_{ij}(\mathbf{q}) : \phi_{ij} - f_{ij}(\mathbf{q}) > 0 \right\} \right\}.$$

Whenever we change the demand on an edge, this is done by an additive δ amount. Reducing flow at most k times to obtain $f_{ij}(\mathbf{q}')$ we guarantee that $\phi_{ij} \geq f_{ij}(\mathbf{q}') \geq (1 - \epsilon) f_{ij}(\mathbf{q})$ holds which implies that the total throughput cannot change by more than a factor $(1 - \epsilon)$.

As we assume that our underlying graph with edge-set $\{(i, j) : \phi_{ij} > 0\}$ is strongly connected, it must be the case that the graph with edge set $\{(i, j) : \phi_{ij} q_{ij} > 0\}$ contains a minimal sequence of components $C_1, C_2, \ldots, C_d = C_1$, $d > 2$, and nodes $u_\ell, v_\ell \in C_\ell$ such that $\phi_{u_\ell v_{\ell+1}} > 0$, but $f_{u_\ell v_{\ell+1}} = 0$. In particular, it being minimal implies that no component other than the first appears repeatedly. Since each u_ℓ, v_ℓ are in the same strongly connected component of the graph with edge-set $\{(i, j) : f_{ij}(\mathbf{q}) > 0\}$, we know that for each ℓ there exists a simple path from u_ℓ to v_ℓ with positive demand on it. We change the quantiles as follows: for all pairs $(u_\ell, v_{\ell+1})$ we increase the quantiles $q_{u_\ell, v_{\ell+1}}$ so that the steady-state rate of units increases by δ and for each edge along the path from u_ℓ to v_ℓ we decrease the quantiles so that the steady-state rate of units decreases by δ. At all other edges the quantiles remain unchanged.

We first argue that this again gives rise to a demand circulation: Each node along a path within a component has its in-flow and out-flow (of demand) reduced by δ, whereas at the nodes u_i, v_i both the sum of in-flows and the sum of out-flows have remained the same. At all other nodes, nothing is altered. Thus, flow conservation continues to hold. By choice of δ none of the edge-capacities are violated. Thus, the resulting demand $\{phi_{ij}q_{ij}\}$ is in the flow relaxation with at most $k-1$ distinct components. Applying this procedure repeatedly, we obtain a single strongly connected component such that the throughput with $\mathbf{q}'$ in the infinite-unit limit (by Lemma 5.3) is within $(1-\epsilon)$ of the throughput of $f_{ij,\infty}(\mathbf{q})$. ■

APPENDIX E

SUPPLEMENTARY MATERIAL FROM CHAPTER 6

E.1 Smoothness of first-price auctions with discrete bids

Before providing the proof of smoothness with discrete bidding space, we formally define a submodular valuations. A valuation function $v_i^t(\cdot)$ is submodular if, for any S, T such that $S \subseteq T$ and any $x \notin T$, $v_i^t(S + \{x\}) - v_i(S) \geq v_i^t(T + \{x\})$.

Lemma 6.2. The simultaneous first price mechanism where players are restricted to bid on at most d items and on each item submit a bid that is a multiple of $\delta \cdot \rho$, is a solution-based $\left(\frac{1}{2} - \delta, 1\right)$-smooth mechanism, when players have submodular valuations, such that all marginals are either 0 or at least ρ and such that each player wants at most d items, i.e. $v_i^t(S) = \max_{T \subseteq S : |T| = d} v_i^t(T)$.

Proof. Consider a valuation profile $v = (v_1, \ldots, v_n)$ for the n players and a bid profile $b = (b_1, \ldots, b_n)$. Each valuation v_i is submodular and thereby also falls into the class of XOS valuations [88], i.e. it can be expressed as a maximum over additive valuations. More formally, for some index set $\mathcal{L}_i$:

$$v_i(S) = \max_{\ell \in \mathcal{L}_i} \sum_{j \in S} a_{ij}^{\ell}$$

Furthermore, by the assumption that marginals are either 0 or at least ρ, it can be easily shown that a_{ij}^{ℓ} is either 0 or at least ρ. Moreover, when the player has value for at most d types of items, it can also be shown that for any $\ell \in \mathcal{L}_i$ at most d of the $(a_{ij}^{\ell})_{j \in [m]}$ will be non-zero.

Consider a feasible allocation $x = (x_1, \ldots, x_n)$ of the items to the bidders, where x_i is the set of types of items allocated to player i (the latter is feasible if each item is never allocated more than its supply). Consider the following deviation $b_i^\star(v_i, x_i)$ that is related to the valuation v_i of player i and to allocation x_i: Let $\ell^\star(x_i) = \arg\max_{\ell \in \mathcal{L}_i} \sum_{j \in x_i} a_{ij}^{\ell}$. Then on each item $j \in x_i$ with $a_{ij}^{\ell^*(x_i)} > 0$, submit $\left\lfloor \frac{a_{ij}^{\ell^*(x_i)}}{2} \right\rfloor_{\delta \cdot \rho}$.[1] On each $j \notin x_i$, submit a zero bid. This submits at most d non-zero bids.

Now we argue that these deviations imply the solution-based smoothness. Let $p_j(b)$ be the lowest winning bid on item j, under bid profile b. Observe that for each j, if $p_j(b) < \left\lfloor \frac{a_{ij}^{\ell^*(x_i)}}{2} \right\rfloor_{\delta \cdot \rho}$, the player wins item j and pays $\left\lfloor \frac{a_{ij}^{\ell^*(x_i)}}{2} \right\rfloor_{\delta \cdot \rho}$. Hence:

$$\begin{aligned}
u_i(b_i^\star(v_i, x_i), b_{-i}; v_i) &\geq \sum_{j \in x_i} \left(a_{ij}^{\ell^\star(x_i)} - \left\lfloor \frac{a_{ij}^{\ell^\star(x_i)}}{2} \right\rfloor_{\delta \cdot \rho} \right) \cdot 1 \left\{ p_j(b) < \left\lfloor \frac{a_{ij}^{\ell^\star(x_i)}}{2} \right\rfloor_{\delta \cdot \rho} \right\} \\
&\geq \sum_{j \in x_i} \left\lfloor \frac{a_{ij}^{\ell^\star(x_i)}}{2} \right\rfloor_{\delta \cdot \rho} \cdot 1 \left\{ p_j(b) < \left\lfloor \frac{a_{ij}^{\ell^\star(x_i)}}{2} \right\rfloor_{\delta \cdot \rho} \right\} \\
&\geq \sum_{j \in x_i} \left(\left\lfloor \frac{a_{ij}^{\ell^\star(x_i)}}{2} \right\rfloor_{\delta \cdot \rho} - p_j(b) \right) \\
&\geq \sum_{j \in x_i} \left(\frac{a_{ij}^{\ell^\star(x_i)}}{2} - \delta \cdot \rho - p_j(b) \right) \\
&\geq \left(\frac{1}{2} - \delta \right) \sum_{j \in x_i} a_{ij}^{\ell^\star(x_i)} - \sum_{j \in x_i} p_j(b) \\
&= \left(\frac{1}{2} - \delta \right) v_i(x_i) - \sum_{j \in x_i} p_j(b)
\end{aligned}$$

Summing over players and observing that $\textsc{Rev}(b) \geq \sum_{j \in x_i} p_j(b)$, we get the lemma.

■

[1]We denote with $\lfloor x \rfloor_{\delta \cdot \rho}$ the highest multiple of $\delta \cdot \rho$ that is less than or equal to x.

APPENDIX F
SUPPLEMENTARY MATERIAL FROM CHAPTER 7.

F.1 Complete proof for group-unaware algorithms

Theorem 7.1. For all $\alpha < 3/8$, there exists $\epsilon > 0$ such that any group-unaware algorithm that satisfies $\mathbb{E}_{\sigma}\left[ApxReg_{\epsilon,T}(f)\right] = o(T)$ for all $f \in \mathcal{F}$ is α-unfair in composition with respect to false negative rate even for perfectly balanced sequences. In particular, for any group-unaware algorithm that ensures vanishing approximate regret[1], there exists an oblivious adversary for assigning labels such that:

- In expectation, half of the population corresponds to each group.
- For each group, in expectation half of its labels are positive and the other half are negative.
- The false negative rates of the two groups differ by α.

Proof. Consider an instance that consists of two groups $\mathcal{G} = \{A, B\}$, two experts $\mathcal{F} = \{h_n, h_u\}$, and two phases: Phase I and Phase II. Group A is the group we end up discriminating against while group B is boosted by the discrimination with respect to false negative rate. At each round t the groups arrive stochastically with probability $1/2$ each, independent of $\sigma^{1:t-1}$.

The experts output a score value in $\hat{\mathcal{Y}} = [0, 1]$, where score $\hat{y}^t_f \in \hat{\mathcal{Y}}$ can be interpreted as the probability that expert f assigns to label being positive in round t, i.e. $y(t) = +$. The loss function is the expected probability of error given

[1]This requirement is weaker than vanishing regret so the impossibility result applies to vanishing regret algorithms.

by $\ell(\hat{y}, y) = \hat{y} \cdot \mathbf{1}\{y = -\} + (1 - \hat{y}) \cdot \mathbf{1}\{y = +\}$. The two experts are very simple: h_n always predicts negative, i.e. $\hat{y}^t_{h_n} = 0$ for all t, and h_u is an unbiased expert who, irrespective of the group or the label, makes an inaccurate prediction with probability $\beta = 1/4 + \sqrt{\epsilon}$, i.e. $\hat{y}^t_{h_u} = \beta \cdot \mathbf{1}\{y(t) = -\} + (1 - \beta) \cdot \mathbf{1}\{y(t) = +\}$ for all t. Both experts are fair in isolation with respect to both false negative and false positive rates: FNR is 100% for h_n and β for h_u regardless the group, and FPR is 0% for h_n and β for h_u, independent of the group. The instance proceeds in two phases:

1. Phase I lasts for $T/2$ rounds. The adversary assigns negative labels on examples with group context B and assigns a label uniformly at random to examples from group A.
2. In Phase II, there are two plausible worlds:
   (a) if the expected probability the algorithm assigns to expert h_u in Phase I is at least $\mathbb{E}_\sigma\left[\sum_{t=1}^{T/2} p^t_{h_u}\right] > \sqrt{\epsilon} \cdot T$ then the adversary assigns negative labels for both groups
   (b) else the adversary assigns positive labels to examples with group context B while examples from group A keep receiving positive and negative labels with probability equal to half.

   We will show that for any algorithm with vanishing approximate regret property, i.e. with $ApxReg_{\epsilon,T}(f) = o(T)$, the condition for the first world is never triggered and hence the above sequence is indeed balanced.

We now show why this instance is unfair in composition with respect to false negative rate. The proof involves showing the following two claims:

1. In Phase I, any ϵ-approximate regret algorithm needs to select the negative

expert h_n most of the times to ensure small approximate regret with respect to h_n. This means that, in Phase I (where we encounter half of the positive examples from group A and none from group B), the false negative rate of the algorithm is close to 1.

2. In Phase II, any ϵ-approximate regret algorithm should quickly catch up to ensure small approximate regret with respect to h_u and hence the false negative rate of the algorithm is closer to β. Since the algorithm is group-unaware, this creates a mismatch between the false negative rate of B (that only receives false negatives in this phase) and A (that has also received many false negatives before).

Upper bound on probability of playing h_u in Phase I. We now formalize the first claim by showing that any algorithm with $\mathbb{E}_\sigma\left[\sum_{t=1}^{T/2} p_{h_u}^t\right] > \sqrt{\epsilon} \cdot T$ does not satisfy the approximate regret property. The algorithm suffers an expected loss of β every time it selects expert h_u. On the other hand, when selecting expert h_n, it suffers a loss of 0 for members of group B and an expected loss of $1/2$ for members of group A. Hence, the algorithm's expected loss in the first phase is:

$$\begin{aligned}\mathop{\mathbb{E}}_{\sigma}\left[\sum_{t=1}^{T/2}\sum_{f\in\mathcal{F}} p_f^t \cdot \ell_f^t\right] &= \mathop{\mathbb{E}}_{\sigma}\left[\sum_{t=1}^{T/2} p_{h_u}^t\right]\cdot\beta + \mathop{\mathbb{E}}_{\sigma}\left[\sum_{t=1}^{T/2} p_{h_n}^t \cdot \mathbf{1}_{g(t)=A}\right]\cdot\frac{1}{2}\\ &= \mathop{\mathbb{E}}_{\sigma}\left[\sum_{t=1}^{T/2} p_{h_u}^t\right]\cdot\beta + \left(\frac{T}{2} - \mathop{\mathbb{E}}_{\sigma}\left[\sum_{t=1}^{T/2} p_{h_u}^t\right]\right)\cdot\frac{1}{4}\\ &= \frac{T}{8} + \left(\beta - \frac{1}{4}\right)\cdot\mathop{\mathbb{E}}_{\sigma}\left[\sum_{t=1}^{T/2} p_{h_u}^t\right] = \frac{T}{8} + \sqrt{\epsilon}\cdot\mathop{\mathbb{E}}_{\sigma}\left[\sum_{t=1}^{T/2} p_{h_u}^t\right]\end{aligned}$$

In contrast, the negative expert has, in Phase I, expected loss of:

$$\mathop{\mathbb{E}}_{\sigma}\left[\sum_{t=1}^{T/2} \ell_{h_n}^t\right] = \frac{T}{8}.$$

Therefore, if $\mathbb{E}_\sigma\left[\sum_{t=1}^{T/2} p^t_{h_u}\right] > \sqrt{\epsilon} \cdot T$, the ϵ-approximate regret of the algorithm with respect to h_n is linear to the time-horizon T (and therefore not vanishing) since:

$$\mathbb{E}_\sigma\left[\sum_{t=1}^{T/2} \sum_{f \in \mathcal{F}} p^t_f \cdot \ell^t_f - (1+\epsilon) \sum_{t=1}^{T/2} \ell^t_{h_N}\right] = \frac{T}{8} + \sqrt{\epsilon} \cdot \mathbb{E}_\sigma\left[\sum_{t=1}^{T/2} p^t_{h_u}\right] - (1+\epsilon)\frac{T}{8} > \frac{7\epsilon}{8} \cdot T.$$

Upper bound on probability of playing h_n in Phase II. Regarding the second claim, we first show that $\mathbb{E}_\sigma\left[\sum_{t=T/2+1}^{T} p^t_{h_n}\right] \le 16\sqrt{\epsilon} \cdot T$ for any ϵ-approximate regret algorithm with $\epsilon < 1/16$. The algorithm's expected loss in the second phase is:

$$\mathbb{E}_\sigma\left[\sum_{t=T/2+1}^{T} \sum_{f \in \mathcal{F}} p^t_f \ell^t_f\right] = \mathbb{E}_\sigma\left[\sum_{t=T/2+1}^{T} p^t_{h_n}\right] \cdot \frac{3}{4} + \left(\frac{T}{2} - \mathbb{E}_\sigma\left[\sum_{t=T/2+1}^{T} p^t_{h_n}\right]\right) \cdot \beta.$$

Since, in Phase I, the best case scenario for the algorithm is to always select expert h_n and incur a loss of $T/8$, this implies that for $\epsilon < 1/16$:

$$\begin{aligned}
\mathbb{E}_\sigma\left[\sum_{t=1}^{T} \sum_{f \in \mathcal{F}} p^t_f \ell^t_f\right] &\ge \frac{T}{8} + \frac{T}{2} \cdot \beta + \mathbb{E}_\sigma\left[\sum_{t=T/2+1}^{T} p^t_{h_n}\right] \cdot \left(\frac{3}{4} - \beta\right) \\
&= \frac{\left(1 + 2\sqrt{\epsilon}\right) \cdot T}{4} + \mathbb{E}_\sigma\left[\sum_{t=T/2+1}^{T} p^t_{h_n}\right] \cdot \left(\frac{1}{2} - \sqrt{\epsilon}\right) \\
&> \frac{T}{4} + \mathbb{E}_\sigma\left[\sum_{t=T/2+1}^{T} p^t_{h_n}\right] \cdot \frac{1}{4}.
\end{aligned}$$

On the other hand, the cumulative expected loss of the β-inaccurate expert h_u is

$$\mathbb{E}\left[\sum_{t=1}^{T} \ell^t_{h_u}\right] = \beta \cdot T = \frac{T}{4} + \sqrt{\epsilon} \cdot T.$$

Therefore, if the algorithm has $\mathbb{E}_\sigma\left[\sum_{t=T/2+1}^{T} p^t_{h_n}\right] > 16\sqrt{\epsilon} \cdot T$, the ϵ-approximate regret of the algorithm with respect to h_u is linear to the time-horizon since:

$$\begin{aligned}
\mathbb{E}_\sigma\left[\sum_{t=1}^{T} \sum_{f \in \mathcal{F}} p^t_f \ell^t_f - (1+\epsilon) \sum_{t=1}^{T} \ell^t_{h_u}\right] &> \left(\frac{T}{4} + \mathbb{E}_\sigma\left[\sum_{t=T/2+1}^{T} p^t_{h_n}\right] \cdot \frac{1}{4}\right) - (1+\epsilon) \cdot \left(\frac{T}{4} + \sqrt{\epsilon} \cdot T\right) \\
&\ge \mathbb{E}_\sigma\left[\sum_{t=T/2+1}^{T} p^t_{h_n}\right] \cdot \frac{1}{4} - 3\sqrt{\epsilon} \cdot T > \sqrt{\epsilon} \cdot T.
\end{aligned}$$

The last inequality holds since $\epsilon \cdot T/4 + \epsilon \cdot \sqrt{\epsilon} \cdot T + \sqrt{\epsilon} \cdot T \leq 3\sqrt{\epsilon} \cdot T$ for $\epsilon \leq 1$.

Thus, we have shown that, when $\epsilon < 1/16$, for any algorithm with vanishing approximate regret, necessarily we have $\mathbb{E}_\sigma\left[\sum_{t=T/2+1}^{T} p^t_{h_n}\right] \leq 16\sqrt{\epsilon} \cdot T$.

Gap in false negative rates between groups *A* and *B*. We now compute the expected false negative rates for the two groups, assuming that $\epsilon < 1/16$. Since we focus on algorithms that satisfy the vanishing regret property, we have already established that:

$$\mathop{\mathbb{E}}_{\sigma}\left[\sum_{t=1}^{T/2} p^t_{h_u}\right] \leq \sqrt{\epsilon} \cdot T \quad \text{and} \quad \mathop{\mathbb{E}}_{\sigma}\left[\sum_{t=T/2+1}^{T} p^t_{h_n}\right] \leq 16\sqrt{\epsilon} \cdot T. \tag{F.1}$$

For ease of notation, let $G^t_{A,+} = \mathbf{1}\{g(t) = A, y(t) = +\}$ and $G^t_{B,+} = \mathbf{1}\{g(t) = B, y(t) = +\}$. Since the group context at round t arrives independent of $\sigma^{1:t-1}$ and the adversary is oblivious, we have that $G^t_{A,+}, G^t_{B,+}$ are independent of $\sigma^{1:t-1}$, and hence also independent of $p^t_{h_u}, p^t_{h_n}$.

Since the algorithm is group-unaware, the expected cumulative probability that the algorithm uses h_n in Phase II is the same for both groups. We combine this with the facts that under the online learning protocol with group context, examples of group B arrive stochastically with probability half but only receive positive labels in Phase II, we obtain:

$$\mathop{\mathbb{E}}_{\sigma}\left[\sum_{t=T/2+1}^{T} p^t_{h_n} \cdot G^t_{B,+}\right] = \frac{1}{2} \cdot \mathop{\mathbb{E}}_{\sigma}\left[\sum_{t=T/2+1}^{T} p^t_{h_n}\right] \leq 8\sqrt{\epsilon} \cdot T. \tag{F.2}$$

Recall that group B in Phase I has no positive labels, hence the false negative rate on group B is:

$$\mathop{\mathbb{E}}_{\sigma}\left[FNR^{\sigma}_{\mathcal{L}}(B)\right] = \mathop{\mathbb{E}}_{\sigma}\left[\frac{\sum_{t=T/2+1}^{T} G^t_{B,+} \cdot \left(p^t_{h_u} \cdot \beta + p^t_{h_n} \cdot 1\right)}{\sum_{t=T/2+1}^{T} \cdot G^t_{B,+}}\right] = \beta + \mathop{\mathbb{E}}_{\sigma}\left[\frac{(1-\beta) \cdot \sum_{t=T/2+1}^{T} G^t_{B,+} \cdot p^t_{h_n}}{\sum_{t=T/2+1}^{T} G^t_{B,+}}\right]$$

In order to upper bound the above false negative rate, we denote the following good event by

$$\mathcal{E}^B(\eta) = \left\{\sigma^{1:T} : \sum_{t=T/2+1}^{T} G^t_{B,+} > (1-\eta)\,\mathbb{E}\left[\sum_{t=T/2+1}^{T} G^t_{B,+}\right]\right\}.$$

By Chernoff bound, the probability of the bad event is:

$$\mathbb{P}\left[\neg\mathcal{E}^B(\eta)\right] = \exp\left(-\frac{\eta^2 \cdot \mathbb{E}\left[\sum_{t=T/2+1}^{T} G^t_{B,+}\right]}{2}\right).$$

For $\eta^B = \sqrt{\frac{16\log(T)}{T}}$, this implies that $\mathbb{P}[\neg\mathcal{E}^B(\eta^B)] \le 1/T^2$ since $\mathbb{E}_\sigma[\sum_{t=T/2+1}^{T} G^t_{B,+}] = \frac{T}{4}$.

Therefore, by first using the bound on $\sum_{t=T/2+1}^{T} G^t_{B,+}$ on the good event and the bound on the probability of the bad event, and then taking the limit $T \to \infty$, it holds that:

$$\begin{aligned}
\mathbb{E}_\sigma\left[FNR^\sigma_{\mathcal{L}}(B)\right] &= \beta + \mathbb{E}_\sigma\left[\frac{(1-\beta)\cdot\sum_{t=T/2+1}^{T} G^t_{B,+}\cdot p^t_{h_n}}{\sum_{t=T/2+1}^{T} G^t_{B,+}}\right] \\
&\le \beta + \frac{1-\beta}{1-\eta^B}\cdot\frac{8\sqrt{\epsilon}\cdot T}{T/4}\cdot\mathbb{P}\left[\mathcal{E}^B(\eta^B)\right] + 1\cdot\mathbb{P}\left[\neg\mathcal{E}^B(\eta^B)\right] \\
&\le \beta + \frac{32\sqrt{\epsilon}}{1-\eta^B} + \frac{1}{T^2} \to \frac{1}{4} + 33\sqrt{\epsilon}.
\end{aligned}$$

We now move to the false negative rate of A:

$$\mathbb{E}_\sigma\left[FNR^\sigma_{\mathcal{L}}(A)\right] = \mathbb{E}_\sigma\left[\frac{\sum_{t=1}^{T} G^t_{A,+}\cdot\left(p^t_{h_u}\cdot\beta + p^t_{h_n}\cdot 1\right)}{\sum_{t=1}^{T} G^t_{A,+}}\right].$$

Letting $\mathcal{E}^A(\eta) = \left\{\sigma^{1:T} : \sum_{t=1}^{T} G^t_{A,+} < (1+\eta)\,\mathbb{E}\left[\sum_{t=1}^{T} G^t_{A,+}\right]\right\}$ and, since $\mathbb{P}[\neg\mathcal{E}^A(\eta)] = \exp\left(-\frac{\eta^2\cdot\mathbb{E}\left[\sum_{t=1}^{T} G^t_{A,+}\right]}{3}\right)$, we obtain that, for $\eta^A = \sqrt{\frac{24\log(T)}{T}}$, $\mathbb{P}[\neg\mathcal{E}^A(\eta^A)] = \frac{1}{T^2}$.

Recall that for our instance $\mathbb{E}_\sigma\left[G^t_{A,+}\right] = T/4$ and $G^t_{A,+}$ is independent of $p^t_{h_u}$. From our previous analysis we also know that:

$$\mathbb{E}_\sigma\left[\sum_{t=1}^{T/2} p^t_{h_u} G^t_{A,+}\right] \le \frac{\sqrt{\epsilon}\cdot T}{4} \quad \text{and} \quad \mathbb{E}_\sigma\left[\sum_{t=T/2+1}^{T} p^t_{h_u} G^t_{A,+}\right] \le \frac{T}{8} \qquad \text{(F.3)}$$

As a result, using that $\mathbb{E}\Big[\sum_{t=1}^{T/2} G_{A,+}^t\Big] = \mathbb{E}\Big[\sum_{t=T/2+1}^{T} G_{A,+}^t\Big] = \frac{T}{8}$ and Inequalities (F.3), we obtain:

$$\mathop{\mathbb{E}}_{\sigma}\left[\sum_{t=1}^{T} G_{A,+}^t \cdot \left(p_{h_u}^t \cdot \beta + p_{h_n}^t \cdot 1\right)\right] = \mathop{\mathbb{E}}_{\sigma}\left[\sum_{t=1}^{T} G_{A,+}^t \cdot - \sum_{t=1}^{T} G_{A,+}^t \cdot p_{h_u}^t(1-\beta)\right]$$
$$\geq \frac{T}{4}\left(1 - (1-\beta)\cdot(\frac{1}{2} + \sqrt{\epsilon})\right).$$

Therefore, similarly with before, it holds that:

$$\mathop{\mathbb{E}}_{\sigma}\Big[FNR_{\mathcal{L}}^{\sigma}(A)\Big] = \mathop{\mathbb{E}}_{\sigma}\left[\frac{\sum_{t=1}^{T} G_{A,+}^t \cdot \left(p_{h_u}^t \cdot \beta + p_{h_n}^t \cdot 1\right)}{\sum_{t=1}^{T} G_{A,+}^t}\right]$$
$$\geq \frac{1-(1-\beta)\cdot(\frac{1}{2}+\sqrt{\epsilon})}{(1+\eta^A)} \cdot \mathbb{P}\Big[\mathcal{E}^A(\eta^A)\Big] + 0 \cdot \mathbb{P}\Big[\neg\mathcal{E}^A(\eta^A)\Big]$$
$$\geq \frac{\frac{1}{2} - \sqrt{\epsilon} + \frac{\beta}{2}}{1+\eta^A}\left(1 - \frac{1}{T^2}\right) > \frac{\frac{1}{2} - \sqrt{\epsilon} + \frac{1}{8}}{1+\eta^A}\left(1 - \frac{1}{T^2}\right) \to \frac{5}{8} - \sqrt{\epsilon}.$$

As a result, the difference between the false negative rate in group A and the one at group B is $\frac{3}{8} + 34\sqrt{\epsilon}$ which can go arbitrarily close to 3/8 by appropriately selecting ϵ to be small enough, for any vanishing approximate regret algorithm. This concludes the proof. ■

F.2 Complete proof for group-aware algorithms

Theorem 7.2. For any group imbalance $b < 0.49$ and $0 < \alpha < \frac{0.49-0.99b}{1-b}$, there exists $\epsilon_0 > 0$ such that for all $0 < \epsilon < \epsilon_0$ any algorithm that satisfies $\mathbb{E}_{\sigma}\Big[ApxReg_{\epsilon,T}(f)\Big] = o(T)$ for all $f \in \mathcal{F}$, is α-unfair in composition.

Proof. The instance has two groups: $\mathcal{G} = \{A, B\}$. Examples with group context A are discriminated against and arrive randomly with probability $b < \frac{1}{2}$ while examples with group context B are boosted by the discrimination and arrive with the remaining probability $1-b$. There are again two experts $\mathcal{F} = \{h_n, h_p\}$, which output score values in $\hat{\mathcal{Y}} = [0, 1]$, where $\hat{y}_f^t$ can be interpreted as the probability that expert f assigns to label being + in round t. We use the earlier loss function of $\ell(\hat{y}, y) = \hat{y} \cdot \mathbf{1}\{y = -\} + (1 - \hat{y}) \cdot \mathbf{1}\{y = +\}$. The first expert h_n is again pessimistic and always predicts negative, i.e. $\hat{y}_{h_n}^t = 0$, while the other expert h_p is optimistic and always predicts positive, i.e. $\hat{y}_{h_p}^t = 1$. These experts again satisfy fairness in isolation with respect to false negative and false positive rate. Let $c = 1/101^2$ denote the percentage of the input that is about positive examples for A, ensuring that $|\mathcal{S}_g^\sigma(FNR)| = \Omega(T)$. The instance proceeds in two phases.

1. Phase I lasts $\Theta \cdot T$ rounds for $\Theta = 101c$. The adversary assigns negative labels on examples with group context B. For examples with group context A, the adversary acts as following:
   - if the algorithm assigns probability on the negative expert below $\gamma(\epsilon) = \frac{99-2\epsilon}{100}$, i.e. $p_{h_n}^t(\sigma^{1:t-1}) < \gamma(\epsilon)$, the adversary assigns negative label.
   - otherwise, the adversary assigns positive labels.
2. In Phase II, there are two plausible worlds:
   (a) the adversary assigns negative labels to both groups if the expected number of times that the algorithm selected the negative expert with probability higher than $\gamma(\epsilon)$ on members of group A is less than $c \cdot b \cdot T$, i.e. $\mathbb{E}_\sigma\Big[\mathbf{1}\Big\{t \le \Theta \cdot T : g(t) = A, p_{h_n}^t \ge \gamma(\epsilon)\Big\}\Big] < c \cdot b \cdot T$.
   (b) otherwise she assigns positive labels to examples with group context B and negative labels to examples with group context A.

   Note that, as before, the condition for the first world will never be triggered by any no-regret learning algorithm (we elaborate on that below) which ensures that $\mathbb{E}_\sigma |S_A^\sigma(FNR)| \ge c \cdot b \cdot T$.

The proof is based on the following claims:

1. In Phase I, any vanishing approximate regret algorithm enters the second world of Phase II.
2. This implies a lower bound on the false negative rate on A, i.e. $FNR(A) \geq \gamma(\epsilon) = \frac{99-2\epsilon}{100}$.
3. In Phase II, any ϵ-approximate regret algorithm assigns large enough probability to the positive expert h_p for group B. This implies an upper bound on the false negative rate on B, i.e. $FNR(B) \leq \frac{1}{2(1-b)}$. Therefore this provides a gap in the false negative rates of at least α.

Proof of first claim. To prove the first claim, we apply the method of contradiction. Assume that the algorithm has $\mathbb{E}_{\sigma}\Big[\mathbf{1}\Big\{t \leq \Theta \cdot T : g(t) = A, p^t_{h_n} \geq \gamma(\epsilon)\Big\}\Big] < c \cdot b \cdot T$. This means that the algorithm faces an expectation of at least $(\Theta - c) \cdot b \cdot T$ negative examples, while predicting the negative expert with probability at most $\gamma(\epsilon) = \frac{99-2\epsilon}{100}$,thereby making an error with probability $1 - \gamma(\epsilon)$. Therefore the expected loss of the algorithm is at least:

$$\mathbb{E}_{\sigma}\left[\sum_{t=1}^{\Theta \cdot T} \sum_{f \in \mathcal{F}} p^t_f \cdot \ell^t_f\right] > (\Theta - c) \cdot b \cdot T \cdot (1 - \gamma(\epsilon)) = c \cdot b \cdot (1 + 2\epsilon) \cdot T.$$

At the same time, expert h_n makes at most $c \cdot b \cdot T$ errors (at the positive examples)

$$\mathbb{E}_{\sigma}\left[\sum_{t=1}^{T} \ell^t_{h_n}\right] < c \cdot b \cdot T.$$

Therefore, if $\mathbb{E}_{\sigma}\Big[\mathbf{1}\{t \leq \Theta \cdot T : g(t) = A, p^t_{h_n} \geq f(\epsilon)\}\Big] < c \cdot b \cdot T$, the ϵ-approximate regret of the algorithm with respect to h_n is linear to the time-horizon T (and therefore not vanishing) since:

$$\mathbb{E}_{\sigma}\left[\sum_{t=1}^{T} \sum_{f \in \mathcal{F}} p^t_f \ell^t_f - (1 + \epsilon) \sum_{t=1}^{T} \ell^t_{h_n}\right] > \epsilon \cdot c \cdot b \cdot T.$$

This violates the vanishing approximate regret property, leading to contradiction.

Proof of second claim. The second claim follows directly by the above construction, since positive examples only appear in Phase I when the probability of error on them is greater than $\gamma(\epsilon)$.

Proof of third claim. Having established that any vanishing approximate regret algorithm will always enter the second world, we now focus on the expected loss of expert h_p in this case. This expert makes errors at most in all Phase I and in the examples of Phase II with group context A:

$$\mathop{\mathbb{E}}_{\sigma}\left[\sum_{t=1}^{T} \ell_{h_p}^t\right] \leq \Theta \cdot T + b \cdot (1-\Theta) \cdot T \leq \Theta \cdot T + 0.49 \cdot (1-\Theta) \cdot T$$

Since group B has only positive examples in Phase II, the expected loss of the algorithm is at least:

$$\mathop{\mathbb{E}}_{\sigma}\left[\sum_{t=1}^{T} \sum_{f \in \mathcal{F}} p_f^t \ell_f^t\right] \geq \mathop{\mathbb{E}}_{\sigma}\left[\sum_{t=\Theta \cdot T+1}^{T} p_{h_n}^t \cdot \mathbf{1}_{g(t)=B}\right]$$

We now show that $\mathbb{E}_{\sigma}\left[\sum_{t=\Theta \cdot T+1}^{T} p_{h_n}^t \cdot \mathbf{1}_{g(t)=B}\right] \leq (\frac{1}{2}+\epsilon) \cdot (1-\Theta) \cdot T$. If this is not the case, then the algorithm does not have vanishing ϵ-approximate regret with respect to expert h_p since:

$$\begin{aligned}\mathop{\mathbb{E}}_{\sigma}\left[\sum_{t=1}^{T} \sum_{f \in \mathcal{F}} p_f^t \ell_f^t - (1+\epsilon) \sum_{t=1}^{T} \ell_{h_p}^t\right] &> \left(\frac{1}{2}+\epsilon\right)(1-\Theta)T - (1+\epsilon)0.49(1-\Theta)T - (1+\epsilon)\Theta T \\ &\geq \left(\frac{1}{2}+\epsilon-0.49-0.49\epsilon\right) \cdot (1-\Theta) \cdot T - (1+\epsilon) \cdot \Theta \cdot T \\ &> (0.01+0.51\epsilon) \cdot \frac{100}{101} \cdot T - \frac{1+\epsilon}{101} \cdot T \geq \frac{50}{101}\epsilon \cdot T\end{aligned}$$

Given the above, we now establish a gap in the fairness with respect to false negative rate. Since group A only experiences positive examples when expert h_n is offered probability higher than $\gamma(\epsilon) = \frac{99-2\epsilon}{100}$, this means that:

$$\mathop{\mathbb{E}}_{\sigma}\left[FNR_{\mathcal{L}}^{\sigma}(A)\right] \to 0.99 - 0.02\epsilon$$

Regarding group B, we need to take into account the low-probability event that the actual realization has significantly less than $(1-b)(1-\Theta)\cdot T$ examples of group B in Phase II (all are positive examples). This can be handled via similar Chernoff bounds as in the proof of the previous theorem. As a result, the expected false negative rate at group B is:

$$\mathop{\mathbb{E}}_{\sigma}\left[FNR_{\mathcal{L}}^{\sigma}(B)\right] \rightarrow \frac{\mathbb{E}_{\sigma}\left[\sum_{t=\Theta\cdot T+1}^{T} p_{h_n}^{t} \cdot \mathbf{1}_{g(t)=B}\right]}{\mathbb{E}_{\sigma}\left[\sum_{t=\Theta\cdot T+1}^{T} \cdot \mathbf{1}_{g(t)=B}\right]} \leq \frac{\left(\frac{1}{2}+\epsilon\right)\cdot(1-\Theta)\cdot T}{(1-b)\cdot(1-\Theta)\cdot T} = \frac{\frac{1}{2}+\epsilon}{1-b}$$

which establishes a gap in the fairness with respect to false negative rate of $\alpha \rightarrow \frac{0.49-0.99b}{1-b}$ selecting $\epsilon > 0$ appropriately small. ■

BIBLIOGRAPHY

[1] Dimitris Achlioptas, Marek Chrobak, and John Noga. Competitive analysis of randomized paging algorithms. *Theor. Comput. Sci.*, 234(1-2):203–218, 2000.

[2] Alekh Agarwal, Daniel Hsu, Satyen Kale, John Langford, Lihong Li, and Robert Schapire. Taming the monster: A fast and simple algorithm for contextual bandits. In *International Conference on Machine Learning (ICML)*, 2014.

[3] Alekh Agarwal, Akshay Krishnamurthy, John Langford, Haipeng Luo, and Robert E. Schapire. Open problem: First-order regret bounds for contextual bandits. In *Proceedings of the 31st Conference on Learning Theory (COLT)*, 2017.

[4] Shipra Agrawal and Navin Goyal. Near-optimal regret bounds for thompson sampling. *Journal of the ACM (JACM)*, 2017.

[5] Chamy Allenberg, Peter Auer, László Györfi, and György Ottucsák. Hannan consistency in on-line learning in case of unbounded losses under partial monitoring. In *Algorithmic Learning Theory (ALT)*, 2006.

[6] Noga Alon, Nicolo Cesa-Bianchi, Claudio Gentile, Shie Mannor, Yishay Mansour, and Ohad Shamir. Nonstochastic multi-armed bandits with graph-structured feedback. *SIAM Journal on Computing*, 2017.

[7] Julia Angwin, Jeff Larson, Surya Mattu, and Lauren Kirchner. Machine bias: Theres software used across the country to predict future criminals. And its biased against blacks. *ProPublica*, 2016.

[8] Julia Angwin and Terry Parris Jr. Facebook lets advertisers exclude users by race. *ProPublica blog*, 28, 2016.

[9] Sanjeev Arora, Elad Hazan, and Satyen Kale. The multiplicative weights update method: a meta-algorithm and applications. *Theory of Computing*, 8(1):121–164, 2012.

[10] Jean-Yves Audibert, Sébastien Bubeck, and Gábor Lugosi. Regret in online combinatorial optimization. *Mathematics of Operations Research*, 2013.

[11] Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Mach. Learn.*, 2002.

[12] Peter Auer, Nicolò Cesa-Bianchi, Yoav Freund, and Robert E. Schapire. The nonstochastic multiarmed bandit problem. *SIAM J. Comput.*, 2003.

[13] Peter Auer, Nicolo Cesa-Bianchi, and Claudio Gentile. Adaptive and self-confident on-line learning algorithms. *Journal of Computer and System Sciences*, 2002.

[14] Peter Auer and Chao-Kai Chiang. An algorithm with nearly optimal pseudo-regret for both stochastic and adversarial bandits. In *Conference on Learning Theory (COLT)*, 2016.

[15] Baruch Awerbuch and Robert D Kleinberg. Adaptive routing with end-to-end feedback: Distributed learning and geometric approaches. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing (STOC)*, 2004.

[16] Santiago R Balseiro, David B Brown, and Chen Chen. Dynamic pricing of relocating resources in large networks. In *Abstracts of the 2019 SIGMET-*

RICS/Performance Joint International Conference on Measurement and Modeling of Computer Systems, pages 29–30. ACM, 2019.

[17] Siddhartha Banerjee, Daniel Freund, and Thodoris Lykouris. Pricing and optimization in shared vehicle systems: An approximation framework. In *Proceedings of the 18th ACM Conference on Economics and Computation (EC)*, 2017.

[18] Nikhil Bansal, Niv Buchbinder, and Joseph (Seffi) Naor. A primal-dual randomized algorithm for weighted paging. *J. ACM*, 2012.

[19] Solon Barocas and Andrew D. Selbst. Big Data's Disparate Impact. *California Law Review*, 2016.

[20] Hamsa Bastani, Mohsen Bayati, and Khashayar Khosravi. Mostly exploration-free algorithms for contextual bandits. *arXiv preprint arXiv:1704.09011*, 2017.

[21] Yahav Bechavod, Katrina Ligett, Aaron Roth, Bo Waggoner, and Zhiwei Steven Wu. Equal opportunity in online classification with partial feedback. *arXiv preprint arXiv:1902.02242*, 2019.

[22] L. A. Belady. A study of replacement algorithms for a virtual-storage computer. *IBM Syst. J.*, 5(2):78–101, June 1966.

[23] Alina Beygelzimer, John Langford, Lihong Li, Lev Reyzin, and Robert E. Schapire. Contextual bandit algorithms with supervised learning guarantees. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2011.

[24] Sarah Bird, Solon Barocas, Kate Crawford, and Hanna Wallach. Exploring or exploiting? social and ethical implications of autonomous experimen-

tation in ai. In *Workshop on Fairness, Accountability, and Transparency in Machine Learning (FAT-ML), New York University*, October 2016.

[25] David Blackwell et al. An analog of the minimax theorem for vector payoffs. *Pacific Journal of Mathematics*, 6(1):1–8, 1956.

[26] Avrim Blum. Empirical support for winnow and weighted-majority algorithms: Results on a calendar scheduling domain. *Machine Learning*, 26(1):5–23, 1997.

[27] Avrim Blum, Eyal Even-Dar, and Katrina Ligett. Routing without regret: On convergence to nash equilibria of regret-minimizing algorithms in routing games. In *Proceedings of the 25th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, 2006.

[28] Avrim Blum, Suriya Gunasekar, Thodoris Lykouris, and Nathan Srebro. On preserving non-discrimination when combining expert advice. In *In Proceedings of the 32nd Advances in Neural Processing Systems (NIPS)*, 2018.

[29] Avrim Blum, MohammadTaghi Hajiaghayi, Katrina Ligett, and Aaron Roth. Regret minimization and the price of total anarchy. In *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing (STOC)*, 2008.

[30] Avrim Blum and Yishay Mansour. From external to internal regret. *Journal of Machine Learning Research*, 2007.

[31] G.W. Brown. Iterative solutions of games by fictitious play. In *Activity Analysis of Production and Allocation*, 1951.

[32] Sébastien Bubeck, Michael B. Cohen, Yin Tat Lee, James R. Lee, and Aleksander Madry. K-server via multiscale entropic regularization. In *Proceed-*

ings of the 50th Annual ACM SIGACT Symposium on Theory of Computing (STOC), 2018.

[33] Sébastien Bubeck, Michael B. Cohen, and Yuanzhi Li. Sparsity, variance and curvature in multi-armed bandits. In *29th International Conference on Algorithmic Learning Theory (ALT)*, 2018.

[34] Sébastien Bubeck and Aleksandrs Slivkins. The best of both worlds: Stochastic and adversarial bandits. In *Conference on Learning Theory (COLT)*, 2012.

[35] Sbastien Bubeck and Nicol Cesa-Bianchi. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends in Machine Learning*, 2012.

[36] Niv Buchbinder and Joseph (Seffi) Naor. The design of competitive online algorithms via a primal: Dual approach. *Found. Trends Theor. Comput. Sci.*, 3(2&#8211;3):93–263, February 2009.

[37] Ioannis Caragiannis, Christos Kaklamanis, Panagiotis Kanellopoulos, Maria Kyropoulou, Brendan Lucier, Renato Paes Leme, and va Tardos. Bounding the inefficiency of outcomes in generalized second price auctions. *Journal of Economic Theory*, 156:343 – 388, 2015. Computer Science and Economic Theory.

[38] L Elisa Celis, Sayash Kapoor, Farnood Salehi, and Nisheeth K Vishnoi. An algorithmic framework to control bias in bandit-based personalization. *arXiv preprint arXiv:1802.08674*, 2018.

[39] Nicolo Cesa-Bianchi, Gábor Lugosi, and Gilles Stoltz. Minimizing regret

with label efficient prediction. *IEEE Transactions on Information Theory*, 51(6):2152–2162, 2005.

[40] Alexandra Chouldechova. Fair prediction with disparate impact: A study of bias in recidivism prediction instruments. *Big data*, 5(2):153–163, 2017.

[41] Sam Corbett-Davies and Sharad Goel. The measure and mismeasure of fairness: A critical review of fair machine learning. *arXiv preprint arXiv:1808.00023*, 2018.

[42] Varsha Dani, Thomas P Hayes, and Sham M Kakade. Stochastic linear optimization under bandit feedback. In *Conference on Learning Theory (COLT)*, 2008.

[43] Amit Daniely, Alon Gonen, and Shai Shalev-Shwartz. Strongly adaptive online learning. In *International Conference on Machine Learning*, pages 1405–1411, 2015.

[44] Constantinos Daskalakis, Alan Deckelbaum, and Anthony Kim. Near-optimal no-regret algorithms for zero-sum games. In *Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete Algorithms*, pages 235–254. Society for Industrial and Applied Mathematics, 2011.

[45] Constantinos Daskalakis and Qinxuan Pan. A counter-example to karlin's strong conjecture for fictitious play. In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, pages 11–20. IEEE, 2014.

[46] Amit Datta, Michael Carl Tschantz, and Anupam Datta. Automated experiments on ad privacy settings. *Proceedings on privacy enhancing technologies*, 2015(1):92–112, 2015.

[47] Nikhil R. Devanur, Kamal Jain, and Robert D. Kleinberg. Randomized primal-dual analysis of ranking for online bipartite matching. In *Proceedings of the Twenty-fourth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2013.

[48] C. Dwork and A. Roth. *The Algorithmic Foundations of Differential Privacy*. Foundations and Trends in Theoretical Computer Science. Now Publishers Incorporated, 2014.

[49] Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard Zemel. Fairness through awareness. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference (ITCS)*, 2012.

[50] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Proceedings of the Third Conference on Theory of Cryptography*, TCC'06, pages 265–284, Berlin, Heidelberg, 2006. Springer-Verlag.

[51] Eyal Even-Dar, Shie Mannor, and Yishay Mansour. Action elimination and stopping conditions for the multi-armed bandit and reinforcement learning problems. *Journal of machine learning research*, 2006.

[52] Amos Fiat, Richard M. Karp, Michael Luby, Lyle A. McGeoch, Daniel D. Sleator, and Neal E. Young. Competitive paging algorithms. *J. Algorithms*, 12(4):685–699, December 1991.

[53] Dylan J. Foster, Zhiyuan Li, Thodoris Lykouris, Karthik Sridharan, and Éva Tardos. Learning in games: Robustness of fast convergence. In *Proceedings of the 30th Advances in Neural Processing Systems (NIPS)*, 2016.

[54] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

[55] Yoav Freund, Robert E Schapire, Yoram Singer, and Manfred K Warmuth. Using and combining predictors that specialize. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing (STOC)*, 1997.

[56] Pratik Gajane, Tanguy Urvoy, and Emilie Kaufmann. Corrupt bandits for privacy preserving input. In *29th International Conference on Algorithmic Learning Theory (ALT)*, 2018.

[57] Stephen Gillen, Christopher Jung, Michael Kearns, and Aaron Roth. Online learning with an unknown fairness metric. In *Advances in Neural Information Processing Systems*, pages 2600–2609, 2018.

[58] John C Gittins. Bandit processes and dynamic allocation indices. *Journal of the Royal Statistical Society: Series B (Methodological)*, 41(2):148–164, 1979.

[59] Eyal Gofer and Yishay Mansour. Lower bounds on individual sequence regret. *Machine Learning*, 103(1):1–26, 2016.

[60] William J Gordon and Gordon F Newell. Closed queuing systems with exponential servers. *Operations research*, 15(2):254–265, 1967.

[61] Anupam Gupta, Tomer Koren, and Kunal Talwar. Better algorithms for stochastic bandits with adversarial corruptions. In *Proceedings of the 33rd Conference On Learning Theory (COLT)*, 2019.

[62] Swati Gupta and Vijay Kamble. Individual fairness in hindsight. In *Proceedings of the 2019 ACM Conference on Economics and Computation (EC)*, 2019.

[63] Mohammad Taghi Hajiaghayi, Robert Kleinberg, and David C Parkes. Adaptive limited-supply online auctions. In *Proceedings of the 5th ACM conference on Electronic commerce*, pages 71–80. ACM, 2004.

[64] James Hannan. Approximation to bayes risk in repeated play. *Contributions to the Theory of Games*, 3:97–139, 1957.

[65] Moritz Hardt, Eric Price, and Nati Srebro. Equality of opportunity in supervised learning. In *Advances in neural information processing systems (NIPS)*, 2016.

[66] Jason D Hartline. *Mechanism Design and Approximation*. 2014.

[67] Elad Hazan and Satyen Kale. Better algorithms for benign bandits. *Journal of Machine Learning Research*, 2011.

[68] Elad Hazan and C. Seshadhri. Adaptive algorithms for online decision problems. *Electronic Colloquium on Computational Complexity (ECCC)*, 14(088), 2007.

[69] Mark Herbster and Manfred K Warmuth. Tracking the best expert. *Machine learning*, 32(2):151–178, 1998.

[70] Justin Hsu, Zhiyi Huang, Aaron Roth, Tim Roughgarden, and Zhiwei Steven Wu. Private matchings and allocations. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, STOC '14, pages 21–30, New York, NY, USA, 2014. ACM.

[71] Matthew Joseph, Michael Kearns, Jamie H Morgenstern, and Aaron Roth. Fairness in learning: Classic and contextual bandits. In *Advances in Neural Information Processing Systems (NIPS)*, 2016.

[72] Adam Kalai and Santosh Vempala. Efficient algorithms for online decision problems. *J. Comput. Syst. Sci.*, 2005.

[73] Sampath Kannan, Michael Kearns, Jamie Morgenstern, Mallesh Pai, Aaron Roth, Rakesh Vohra, and Zhiwei Steven Wu. Fairness incentives for myopic agents. In *Proceedings of the 2017 ACM Conference on Economics and Computation*, pages 369–386. ACM, 2017.

[74] Sampath Kannan, Jamie H Morgenstern, Aaron Roth, Bo Waggoner, and Zhiwei Steven Wu. A smoothed analysis of the greedy algorithm for the linear contextual bandit problem. In *Advances in Neural Information Processing Systems*, pages 2227–2236, 2018.

[75] Richard M Karp, Umesh V Vazirani, and Vijay V Vazirani. An optimal algorithm for on-line bipartite matching. In *Proceedings of the twenty-second annual ACM symposium on Theory of computing*, pages 352–358. ACM, 1990.

[76] Michael Kearns, Mallesh M. Pai, Aaron Roth, and Jonathan Ullman. Mechanism design in large games: incentives and privacy. In *Innovations in Theoretical Computer Science, ITCS'14, Princeton, NJ, USA, January 12-14, 2014*, pages 403–410, 2014.

[77] Frank P Kelly. *Reversibility and Stochastic Networks*. Cambridge University Press, 2011.

[78] Jon M. Kleinberg, Sendhil Mullainathan, and Manish Raghavan. Inherent trade-offs in the fair determination of risk scores. In *Innovations of Theoretical Computer Science (ITCS)*, 2017.

[79] Robert Kleinberg, Georgios Piliouras, and Eva Tardos. Multiplicative updates outperform generic no-regret learning in congestion games. In

Proceedings of the forty-first annual ACM symposium on Theory of computing, pages 533–542. ACM, 2009.

[80] Robert Kleinberg, Aleksandrs Slivkins, and Eli Upfal. Multi-armed bandits in metric spaces. In *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing (STOC)*, 2008.

[81] Tomás Kocák, Gergely Neu, Michal Valko, and Rémi Munos. Efficient learning by implicit exploration in bandit problems with side observations. In *28th Annual Conference on Neural Information Processing Systems (NIPS)*, 2014.

[82] Elias Koutsoupias and Christos Papadimitriou. Worst-case equilibria. In *Annual Symposium on Theoretical Aspects of Computer Science*, pages 404–413. Springer, 1999.

[83] Elias Koutsoupias and Christos H. Papadimitriou. On the k-server conjecture. *J. ACM*.

[84] Tim Kraska, Alex Beutel, Ed H Chi, Jeffrey Dean, and Neoklis Polyzotis. The case for learned index structures. In *Proceedings of the 2018 International Conference on Management of Data*, pages 489–504. ACM, 2018.

[85] Ulrich Krengel and Louis Sucheston. Semiamarts and finite values. *Bulletin of the American Mathematical Society*, 83(4):745–747, 1977.

[86] Tze Leung Lai and Herbert Robbins. Asymptotically efficient adaptive allocation rules. *Advances in applied mathematics*, 6(1):4–22, 1985.

[87] John Langford and Tong Zhang. The epoch-greedy algorithm for contextual multi-armed bandits. In *Proceedings of the 20th International Conference on Neural Information Processing Systems (NIPS)*. Citeseer, 2007.

[88] Benny Lehmann, Daniel Lehmann, and Noam Nisan. Combinatorial auctions with decreasing marginal utilities. In *Proceedings of the 3rd ACM Conference on Electronic Commerce*, EC '01, pages 18–28, New York, NY, USA, 2001. ACM.

[89] Lydia T. Liu, Sarah Dean, Esther Rolf, Max Simchowitz, and Moritz Hardt. Delayed impact of fair machine learning. *35th International Conference on Machine Learning (ICML)*, 2018.

[90] Yang Liu, Goran Radanovic, Christos Dimitrakakis, Debmalya Mandal, and David C. Parkes. Calibrated fairness in bandits. *Workshop on Fairness, Accountability, and Transparency in Machine Learning (FAT-ML)*, 2017.

[91] Haipeng Luo and Robert E Schapire. Achieving all with no parameters: Adanormalhedge. In *Conference on Learning Theory (COLT)*, pages 1286–1304, 2015.

[92] Thodoris Lykouris, Vahab Mirrokni, and Renato Paes Leme. Stochastic bandits robust to adversarial corruptions. In *Proceedings of the 50th ACM Annual Symposium on Theory of Computing (STOC)*, 2018.

[93] Thodoris Lykouris, Karthik Sridharan, and Éva Tardos. Small-loss bounds for online learning with partial information. In *Proceedings of the 31st Annual Conference on Learning Theory (COLT)*, 2018.

[94] Thodoris Lykouris, Vasilis Syrgkanis, and Éva Tardos. Learning and efficiency in games with dynamic population. In *Proceedings of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2016.

[95] Thodoris Lykouris and Sergei Vassilvitskii. Competitive caching with

machine learned advice. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, 2018.

[96] Shie Mannor and Ohad Shamir. From bandits to experts: On the value of side-observations. In *Advances in Neural Information Processing Systems (NIPS)*, 2011.

[97] Andrés Muñoz Medina and Sergei Vassilvitskii. Revenue optimization with approximate bid predictions. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017.

[98] Panayotis Mertikopoulos, Christos Papadimitriou, and Georgios Piliouras. Cycles in adversarial regularized learning. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2703–2717. SIAM, 2018.

[99] Michael Mitzenmacher. Scheduling with predictions and the price of misprediction. *arXiv preprint arXiv:1902.00732*, 2019.

[100] Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms*. Cambridge University Press, New York, NY, USA, 1995.

[101] John Nash. Non-cooperative games. *Annals of mathematics*, pages 286–295, 1951.

[102] Denis Nekipelov, Vasilis Syrgkanis, and Eva Tardos. Econometrics for learning agents. In *Proceedings of the Sixteenth ACM Conference on Economics and Computation (EC)*, 2015.

[103] Gergely Neu. Explore no more: Improved high-probability regret bounds for non-stochastic bandits. In *Proceedings of the 28th Annual Conference on Neural Information Processing Systems (NIPS)*, 2015.

[104] Gergely Neu. First-order regret bounds for combinatorial semi-bandits. In *Conference on Learning Theory (COLT)*, 2015.

[105] Gergely Neu and Gábor Bartók. An efficient algorithm for learning with semi-bandit feedback. In *24th International Conference on Algorithmic Learning Theory (ALT)*, 2013.

[106] David C Parkes and Satinder P Singh. An mdp-based approach to online mechanism design. In *Advances in neural information processing systems*, pages 791–798, 2004.

[107] Manish Purohit, Zoya Svitkina, and Ravi Kumar. Improving online algorithms via ml predictions. In *Advances in Neural Information Processing Systems*, pages 9661–9670, 2018.

[108] Pengyu Qian, Siddhartha Banerjee, and Yash Kanoria. The value of state dependent control in ridesharing systems. *arXiv preprint arXiv:1803.04959*, 2018.

[109] Manish Raghavan, Aleksandrs Slivkins, Jennifer Wortman Vaughan, and Zhiwei Steven Wu. The externalities of exploration and how data diversity helps exploitation. In *Conference on Learning Theory (COLT)*, 2018.

[110] Alexander Rakhlin and Karthik Sridharan. Online learning with predictable sequences. In Shai Shalev-Shwartz and Ingo Steinwart, editors, *Proceedings of the 26th Annual Conference on Learning Theory*, volume 30 of *Proceedings of Machine Learning Research*, pages 993–1019, Princeton, NJ, USA, 12–14 Jun 2013. PMLR.

[111] Sasha Rakhlin and Karthik Sridharan. Optimization, learning, and games

with predictable sequences. In *Advances in Neural Information Processing Systems*, pages 3066–3074, 2013.

[112] Frank P Ramsey. A contribution to the theory of taxation. *The Economic Journal*, 37(145):47–61, 1927.

[113] Herbert Robbins. Some aspects of the sequential design of experiments. *Bulletin of the American Mathematical Society*, 58(5):527–535, 1952.

[114] Julia Robinson. An iterative method of solving a game. *Annals of mathematics*, pages 296–301, 1951.

[115] Ryan M. Rogers, Aaron Roth, Jonathan Ullman, and Zhiwei Steven Wu. Inducing approximately optimal flow using truthful mediators. In *Proceedings of the Sixteenth ACM Conference on Economics and Computation*, EC'15. ACM, 2015.

[116] Tim Roughgarden. Intrinsic robustness of the price of anarchy. In *Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing (STOC)*, 2009.

[117] Tim Roughgarden and Éva Tardos. How bad is selfish routing? *J. ACM*, 49(2):236–259, March 2002.

[118] Paat Rusmevichientong and John N Tsitsiklis. Linearly parameterized bandits. *Mathematics of Operations Research*, 35(2):395–411, 2010.

[119] Yevgeny Seldin and Gábor Lugosi. An improved parametrization and analysis of the exp3++ algorithm for stochastic and adversarial bandits. In *Conference on Learning Theory (COLT)*, 2017.

[120] Yevgeny Seldin and Aleksandrs Slivkins. One practical algorithm for both stochastic and adversarial bandits. In *International Conference on Machine Learning (ICML)*, 2014.

[121] Richard Serfozo. Introduction to stochastic networks, 1999.

[122] Latanya Sweeney. Discrimination in online ad delivery. *Queue*, 11(3):10:10–10:29, March 2013.

[123] Vasilis Syrgkanis and Éva Tardos. Composable and efficient mechanisms. In *Proceedings of the 45th Annual Symposium on Theory of Computing (STOC)*, 2013.

[124] W. R. Thompson. On the Likelihood that one Unknown Probability Exceeds Another in View of the Evidence of Two Samples. *Biometrika*, 1933.

[125] Ariel Waserhole and Vincent Jost. Pricing in vehicle sharing systems: optimization in queuing networks with product forms. *EURO Journal on Transportation and Logistics*, pages 1–28, 2014.

[126] Richard Weber et al. On the gittins index for multiarmed bandits. *The Annals of Applied Probability*, 2(4):1024–1033, 1992.

[127] Chen-Yu Wei and Haipeng Luo. More adaptive algorithms for adversarial bandits. In *Proceedings of the 31st Conference On Learning Theory (COLT)*, 2018.