Oracle-Efficient Online Learning and Auction Design*

MIROSLAV DUDÍK, Microsoft Research, New York City
NIKA HAGHTALAB, Cornell University
HAIPENG LUO, University of Southern California
ROBERT E. SCHAPIRE, Microsoft Research, New York City
VASILIS SYRGKANIS, Microsoft Research, New England
JENNIFER WORTMAN VAUGHAN, Microsoft Research, New York City

We consider the design of computationally efficient online learning algorithms in an adversarial setting in which the learner has access to an offline optimization oracle. We present an algorithm called Generalized Follow-the-Perturbed-Leader and provide conditions under which it is oracle-efficient while achieving vanishing regret. Our results make significant progress on an open problem raised by Hazan and Koren [31], who showed that oracle-efficient algorithms do not exist in general [30] and asked whether one can identify properties under which oracle-efficient online learning may be possible.

Our auction-design framework considers an auctioneer learning an optimal auction for a sequence of adversarially selected valuations with the goal of achieving revenue that is almost as good as the optimal auction in hindsight, among a class of auctions. We give oracle-efficient learning results for: (1) VCG auctions with bidder-specific reserves in single-parameter settings, (2) envy-free item pricing in multi-item auctions, and (3) s-level auctions of Morgenstern and Roughgarden [43] for single-item settings. The last result leads to an approximation of the overall optimal Myerson auction when bidders' valuations are drawn according to a fast-mixing Markov process, extending prior work that only gave such guarantees for the i.i.d. setting.

Finally, we derive various extensions, including: (1) oracle-efficient algorithms for the contextual learning setting in which the learner has access to side information (such as bidder demographics), (2) learning with approximate oracles such as those based on Maximal-in-Range algorithms, and (3) no-regret bidding in simultaneous auctions, resolving an open problem of Daskalakis and Syrgkanis [14].

CCS Concepts: • **Theory of computation → Online learning theory**; *Computational pricing and auctions*; Convergence and learning in games.

Additional Key Words and Phrases: online learning, oracle-efficiency, follow-the-perturbed-leader, auction design

ACM Reference Format:
Miroslav Dudík, Nika Haghtalab, Haipeng Luo, Robert E. Schapire, Vasilis Syrgkanis, and Jennifer Wortman Vaughan. 2020. Oracle-Efficient Online Learning and Auction Design. *J. ACM* 1, 1, Article 1 (January 2020), 57 pages. https://doi.org/10.1145/3402203

*An extended abstract of this paper, excluding Sections 3.2 and 4–7 and all the proofs except the proof of Lemma 2.4, appeared in the *Proceedings of the 58th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2017)* [20].

Authors' addresses: Miroslav Dudík, mdudik@microsoft.com, Microsoft Research, New York City; Nika Haghtalab, Cornell University, nika@cs.cornell.edu; Haipeng Luo, University of Southern California, haipengl@usc.edu; Robert E. Schapire, Microsoft Research, New York City, schapire@microsoft.com; Vasilis Syrgkanis, Microsoft Research, New England, vasy@microsoft.com; Jennifer Wortman Vaughan, Microsoft Research, New York City, jenn@microsoft.com.

0004-5411/2020/1-ART1 $15.00
https://doi.org/10.1145/3402203

1 INTRODUCTION

Online learning plays a major role in the adaptive optimization of computer systems, from the design of online marketplaces [3, 5, 9, 50] to the optimization of routing schemes in communication networks [2]. The environments in these applications are constantly evolving, requiring continued adaptation of these systems. Online learning algorithms have been designed to robustly address this challenge, with performance guarantees that hold even when the environment is adversarial. Several standard approaches, including multiplicative weights and exponentiated gradient algorithms [24, 38], work with arbitrary payoff functions and achieve statistically optimal guarantees, but they are computationally inefficient when the learner's action space is exponential in the natural problem representation. For certain action spaces and environments, efficient online learning algorithms can be designed by reducing the online learning problem to an optimization problem [2, 29, 35, 37]. However, these approaches do not easily extend to the complex and highly non-linear problems faced by real learning systems, such as the learning systems used in online market design. In this paper, we address the problem of efficient online learning with an exponentially large action space under arbitrary learner objectives.

This goal is not achievable without some assumptions on the problem structure. Since an online optimization problem is at least as hard as the corresponding offline optimization problem [8, 14], a minimal assumption is the existence of an algorithm that returns a near-optimal solution to the offline problem. We assume that our learner has access to such an offline algorithm, which we call *an offline optimization oracle*. This oracle, for any (weighted) history of choices by the environment, returns an action of the learner that (approximately) maximizes the learner's reward. We seek to design *oracle-efficient learners*, that is, learners that run in polynomial time, with each oracle call counting $O(1)$.

An oracle-efficient learning algorithm can be viewed as a reduction from the online to the offline problem, providing conditions under which the online problem is not only as hard, but also as easy as the offline problem, and thereby offering computational equivalence between online and offline optimization. Apart from theoretical significance, reductions from online to offline optimization are also practically important. For example, if one has already developed and implemented a Bayesian optimization procedure which optimizes against a static stochastic environment, then our algorithm offers a black-box transformation of that procedure into an adaptive optimization algorithm with provable learning guarantees in non-stationary, non-stochastic environments. Even if the existing optimization system does not run in worst-case polynomial time, but is rather a well-performing fast heuristic, a reduction to offline optimization is capable of leveraging any expert domain knowledge that went into designing the heuristic, as well as any further improvements of the heuristic or even discovery of polynomial-time solutions.

Recent work of Hazan and Koren [30] shows that oracle-efficient learning in adversarial environments is not achievable in general, while leaving open the problem of identifying the properties under which oracle-efficient online learning may be possible [31]. We introduce a generic algorithm called *Generalized Follow-the-Perturbed-Leader* (Generalized FTPL) and derive sufficient conditions under which this algorithm yields oracle-efficient online learning. Our results are enabled by providing a new way of adding *regularization* so as to *stabilize* optimization algorithms in general optimization settings. The latter could be of independent interest beyond online learning. Our approach unifies and extends previous approaches to oracle-efficient learning, including the Follow-the-Perturbed Leader (FTPL) approach introduced by Kalai and Vempala [37] for linear objective functions, and its generalizations to submodular objective functions [29], adversarial contextual learning [53], and learning in simultaneous second-price auctions [14]. Furthermore,

our sufficient conditions are related to the notion of a universal identification set of Goldman et al. [25] and oracle-efficient online optimization techniques of Daskalakis and Syrgkanis [14].

The second main contribution of our work is to introduce a new framework for the problem of adaptive auction design for revenue maximization and to demonstrate the power of Generalized FTPL through several applications in this framework. Traditional auction theory assumes that the valuations of the bidders are drawn from a population distribution which is known, thereby leading to a Bayesian optimization problem. The knowledge of the distribution by the seller is a strong assumption. Recent work in algorithmic mechanism design [13, 17, 18, 33, 43, 49] relaxes this assumption by solely assuming access to a set of samples from the distribution. In this work, we drop any distributional assumptions and introduce the adversarial learning framework of *online auction design*. On each round, a learner adaptively designs an auction rule for the allocation of a set of resources to a fresh set of bidders from a population.[1] The goal of the learner is to achieve average revenue at least as large as the revenue of the best auction from some target class. Unlike the standard approach to auction design, initiated by the seminal work of Myerson [44], our approach is devoid of any assumptions about a prior distribution on the valuations of the bidders for the resources at sale. Instead, similar to an agnostic approach in learning theory, we incorporate prior knowledge in the form of a target class of auction schemes that we want to compete with. This is especially appropriate when the auctioneer is restricted to using a particular design of auctions with power to make only a few design choices, such as deciding the reserve prices in a second-price auction. A special case of our framework is considered in the recent work of Roughgarden and Wang [50]. They study online learning of the class of single-item second-price auctions with bidder-specific reserves, and give an algorithm with performance that approaches a constant factor of the optimal revenue in hindsight. We go well beyond this specific setting and show that our Generalized FTPL can be used to optimize over several standard classes of auctions including VCG auctions with bidder-specific reserves and the level auctions of Morgenstern and Roughgarden [43], achieving low additive regret to the best auction in the class.

In the remainder of this section, we describe our main results in more detail and then discuss several extensions and applications of these results, including (1) learning with side information (i.e., contextual learning); (2) learning with constant-factor approximate oracles (e.g., using Maximal-in-Range algorithms [45]); (3) regret bounds with respect to stronger benchmarks for the case in which the environment is not completely adversarial, but follows a fast-mixing Markov process.

Our work contributes to two major research agendas: the design of efficient and oracle-efficient online learning algorithms [1, 14, 21, 29, 30, 35, 37, 48, 54], and the design of auctions using machine learning tools [5, 9, 13, 17, 39, 43]. We describe related work from both areas in more detail in Appendix A.

1.1 Oracle-Efficient Learning with Generalized FTPL

We consider the following online learning problem. On each round $t = 1, \ldots, T$, a learner chooses an action x_t from a finite set $\mathcal{X}$, and an adversary chooses an action y_t from a set $\mathcal{Y}$. The learner then observes y_t and receives a payoff $f(x_t, y_t) \in [0, 1]$, where the function f is fixed and known to the learner. The goal of the learner is to obtain low expected regret with respect to the best action in hindsight, i.e., to minimize

$$\textsc{Regret} := \mathbb{E}\left[\max_{x \in \mathcal{X}} \sum_{t=1}^{T} f(x, y_t) - \sum_{t=1}^{T} f(x_t, y_t)\right],$$

[1] Equivalently, the set of bidders on each round can be the same as long as they are myopic and optimize their utility separately in each round. Using our extension to contextual learning (Section 5), this approach can also be applied when the learner's choice of auction is allowed to depend on features of the arriving set of bidders, such as demographic information.

where the expectation is over the randomness of the learner.[2] We desire algorithms, called *no-regret algorithms*, for which this regret is sublinear in the time horizon T.

In addition, we would like our algorithms to be *oracle-efficient*. By this we mean that they should have an efficient implementation using access to an *offline optimization oracle* that for every set of weighted adversary actions $S = \{(w_\ell, y_\ell)\}_{\ell \in \mathcal{L}}$ with $w_\ell \in \mathbb{R}^+$, $y_\ell \in \mathcal{Y}$ returns an approximately optimal response, that is an action $\hat{x}$ such that

$$\sum_{(w,y)\in S} w f(\hat{x}, y) \geq \max_{x \in \mathcal{X}} \sum_{(w,y)\in S} w f(x, y) - \epsilon,$$

for a desired precision ϵ. For ease of presentation, in the main body of this paper we work with offline optimization oracles that take data with arbitrary non-negative real weights as their input. As we discuss in Section 2.2 and Appendix F, our results extend immediately to weaker oracles, such as those that only take non-negative integer weights.

Our algorithm takes its name from the seminal Follow-the-Perturbed-Leader (FTPL) algorithm of Kalai and Vempala [37]. FTPL achieves low regret, $O(\sqrt{T \log |\mathcal{X}|})$, by independently perturbing the historical payoff of each of the learner's actions and choosing on each round the action with the highest perturbed payoff. However, this approach is inefficient when the action space is exponential in the natural representation of the learning problem, because it requires creating $|\mathcal{X}|$ independent random variables.[3] Moreover, because of the form of the perturbation, the optimization of the perturbed payoffs cannot be performed by the offline optimization oracle for the same problem. We overcome both of these challenges by, first, generalizing FTPL to work with perturbations that can be compactly represented and are thus not necessarily independent across different actions (*sharing randomness*), and, second, by implementing such perturbations via synthetic histories of adversary actions and thus creating offline problems of the same form as the online problem (*implementing randomness*).

Sharing Randomness. Our Generalized FTPL begins by drawing a random vector $\boldsymbol{\alpha} \in \mathbb{R}^N$ of dimension N, with components α_j drawn independently from a distribution D that is not too highly concentrated on any small interval. We call such distributions *dispersed* (see Definition 2.3 below). The payoff of each of the learner's actions is perturbed by a linear combination of α_js, as prescribed by a *perturbation translation matrix* $\boldsymbol{\Gamma}$ of size $|\mathcal{X}| \times N$, with entries in $[0, 1]$. Let $\boldsymbol{\Gamma}_x$ denote the row of $\boldsymbol{\Gamma}$ corresponding to x. On each round t, the algorithm outputs an action x_t that (approximately) maximizes the perturbed historical performance. In other words, x_t is chosen such that for all $x \in \mathcal{X}$,

$$\sum_{\tau=1}^{t-1} f(x_t, y_\tau) + \boldsymbol{\alpha} \cdot \boldsymbol{\Gamma}_{x_t} \geq \sum_{\tau=1}^{t-1} f(x, y_\tau) + \boldsymbol{\alpha} \cdot \boldsymbol{\Gamma}_x - \epsilon$$

for some fixed optimization accuracy $\epsilon \geq 0$. This procedure is fully described in Algorithm 1 of Section 2.

We show that Generalized FTPL is no-regret as long as ϵ is sufficiently small and the translation matrix $\boldsymbol{\Gamma}$ satisfies an *admissibility* condition. This condition requires the rows of $\boldsymbol{\Gamma}$ to be (sufficiently) distinct so that each action's perturbation uses a different weighted combination of the N-dimensional noise. To the best of our knowledge, the approach of using an arbitrary matrix to induce shared randomness among actions of the learner is novel. The formal no-regret result is in Theorem 2.5. The informal statement is the following:

[2]To simplify exposition, we assume that the adversary is oblivious, i.e., that the sequence $y_1, \ldots, y_T$ is chosen in advance. Our results generalize to adaptive adversaries using standard techniques [14, 34].

[3]If payoffs are linear in some low-dimensional representation of $\mathcal{X}$ then the number of variables needed is equal to this dimension. But for non-linear payoffs, $|\mathcal{X}|$ variables are required.

Informal Theorem 1.1. *A translation matrix is δ-admissible if any two rows of the matrix are distinct and the minimum non-zero difference between any two values within a column is at least δ. Generalized FTPL with a δ-admissible matrix $\mathbf{\Gamma}$ and an appropriate distribution D achieves regret $O(N\sqrt{T}/\delta + \epsilon T)$.*

A technical challenge here is to show that the randomness induced by $\mathbf{\Gamma}$ on the set of actions $\mathcal{X}$ *stabilizes* the algorithm, i.e., the probability that $x_t \neq x_{t+1}$ is small. We use the admissibility of $\mathbf{\Gamma}$ to guide us through the analysis of stability. In particular, we consider how each column of $\mathbf{\Gamma}$ partitions actions of $\mathcal{X}$ to a few subsets (at most $1 + \delta^{-1}$) based on their corresponding entries in that column. Since the matrix rows are distinct, the algorithm is stable as a whole if, for each column, the partition to which the algorithm's chosen action belongs remains the same between consecutive rounds with probability close to 1. This allows us to decompose the stability analysis of the algorithm as a whole to the analysis of stability across partitions of each column. At the column level, stability of the partition between two rounds follows by showing that a switch between partitions happens only if the perturbation α_j corresponding to that column falls into a small sub-interval of the support of the distribution D, from which it is sampled. The latter probability is small if D is sufficiently dispersed. This final argument is similar in nature to the reason why perturbations lead to stability in the original FTPL algorithm of Kalai and Vempala [37].

Implementing Randomness. To ensure oracle-efficient learning, we additionally need the property that the induced action-level perturbations can be simulated by a (short) synthetic history of adversary actions. This allows us to avoid working with $\mathbf{\Gamma}$ directly, or even explicitly writing it down. This requirement is captured by our *implementability* condition, which states that each column of the translation matrix corresponds to a scaled version of the expected reward of the learner on some distribution of adversary actions. The formal statement is in Theorem 2.10. The informal statement is the following:

Informal Theorem 1.2. *A translation matrix is implementable if each column corresponds to a scaled version of the expected reward of the learner against some finitely supported distribution of actions of the adversary. Generalized FTPL with an implementable translation matrix can be implemented with one oracle call per round and runs in time polynomial in N, T, and the size of the support of the distributions implementing the translation matrix. Oracle calls count $O(1)$ in the running time.*

The use of synthetic histories in online optimization was first explored by Daskalakis and Syrgkanis [14], who sample histories of length $\text{poly}(|\mathcal{Y}|)$ from a fixed distribution. Our implementability property uses matrix $\mathbf{\Gamma}$ to obtain problem-specific distributions that stabilize online optimization with shorter histories.

For some learning problems, it is easier to first construct an implementable translation matrix and argue about its admissibility; for others, it is easier to construct an admissible matrix and argue about its implementability. We pursue both strategies in various applications, demonstrating the versatility of our conditions.

Our theorems yield the following simple sufficient condition for oracle-efficient no-regret learning (see Theorems 2.5 and 2.10 for more general statements):

> If there exist N adversary actions such that any two actions of the learner yield different rewards on at least one of these N actions, then Generalized FTPL with an appropriate translation matrix has regret $O(N\sqrt{T}/\delta)$ and its oracle-based runtime is $\text{poly}(N, T)$ where δ is the smallest difference between distinct rewards obtainable on any one of the N adversary actions.

The aforementioned results establish a reduction from online optimization to offline optimization. Recall that in the oracle-based runtime, each oracle call counts $O(1)$. When the offline optimization problem can be solved in polynomial time, these results imply that the online optimization problem can also be solved in (fully) polynomial time. The formal statement is in Corollary 2.11.

1.2 Main Application: Online Auction Design

In many applications of auction theory, including electronic marketplaces, a seller repeatedly sells an item or a set of items to a population of buyers, with a few arriving for each auction. In such cases, the seller can optimize his auction design in an online manner, using historical data consisting of observed bids. We consider a setting in which the seller would like to use this historical data to select an auction from a fixed target class. For example, a seller in a sponsored-search auction might be limited by practical constraints to consider only second-price auctions with bidder-specific reserves. The seller can optimize the revenue by using the historical data for each bidder to set these reserves. Similarly, a seller on eBay may be restricted to set a single reserve price for each item. Here, the seller can optimize the revenue by using historical data from auctions for similar goods to set the reserves for new items. In both cases, the goal is to leverage the historical data to pick an auction on each round in such a way that the seller's overall revenue compares favorably with the optimal auction from the target class.

More formally, on round $t = 1, \ldots, T$, a tuple of n bidders arrives with a vector of n bids or, equivalently, a vector of valuations (since we only consider truthful auctions), denoted $\mathbf{v}_t \in \mathcal{V}^n$. We allow these valuations to be arbitrary, e.g., chosen by an adversary. Prior to observing the bids, the auctioneer commits to an auction a_t from a class of truthful auctions $\mathcal{A}$. The goal of the auctioneer is to achieve a revenue that, in hindsight, is very close to the revenue that would have been achieved by the best fixed auction in class $\mathcal{A}$ if that auction were used on all rounds. In other words, the auctioneer aims to minimize the expected regret

$$\mathbb{E}\left[\max_{a\in\mathcal{A}} \sum_{t=1}^{T} \mathrm{Rev}(a, \mathbf{v}_t) - \sum_{t=1}^{T} \mathrm{Rev}(a_t, \mathbf{v}_t)\right],$$

where $\mathrm{Rev}(a, \mathbf{v})$ is the revenue of auction a on bid profile $\mathbf{v}$ and the expectation is over the actions of the auctioneer.

This problem can easily be cast in our oracle-efficient online learning framework. The learner's action space is the set of target auctions $\mathcal{A}$, while the adversary's action space is the set of bid or valuation vectors $\mathcal{V}^n$. The offline oracle is a revenue maximization oracle which computes an (approximately) optimal auction within the class $\mathcal{A}$ for any given set of valuation vectors. Using the Generalized FTPL with appropriate matrices $\mathbf{\Gamma}$, we provide the first oracle-efficient no-regret algorithms for three commonly studied auction classes:

- Vickrey-Clarkes-Groves (VCG) auctions with bidder-specific reserve prices in single-dimensional matroid settings, which are known to achieve half the revenue of the optimal auction under certain conditions, when bidder valuations are independent [28];
- envy-free item-pricing mechanisms in combinatorial markets with unlimited supply, often studied in the static Bayesian setting [3, 26];
- single-item level auctions, introduced by Morgenstern and Roughgarden [43], who show that these auctions approximate, to an arbitrary accuracy, the Myerson auction [44], which is known to be optimal for the Bayesian independent-private-value setting.

The crux of our approach is in designing admissible and implementable matrices for these auction classes. At a high level, a common auction property that makes this possible is the following:

Table 1. Regret bounds and oracle-based runtime for the auction classes considered in this work, for n bidders and time horizon T. All our algorithms perform a single oracle call per round.

Auction Class	Regret	Oracle-Based Runtime	Section
VCG with bidder-specific reserves, s-unit	$O(ns\sqrt{T}\log T)$	$O(T^2 + nT^{3/2}\log T)$	3.1
envy free k-item pricing with infinite supply and unit-demand or single-minded bidders	$O\left(nk\sqrt{T}\log(kT)\right)$	$O\left(T^2 + k^2T^{3/2}\log(kT)\right)$	3.2
level auction[a] with discretization level m	$O(n^3m^4\sqrt{T})$	$O(T^2 + n^3m^3T)$	3.3

[a]In the extended abstract [20] we presented the regret bound $O(nm^2\sqrt{T})$ and running time $O(T^2 + nmT)$, corresponding to discretized level auctions with distinct thresholds (see Theorem 3.13). Here, we present the result that allows repetitions of threshold values (see Theorem 3.15).

> For any pair of auctions a and a' in a given class, there exists a *sparse* bid profile $\mathbf{v}$ such that $\mathrm{Rev}(a,\mathbf{v})$ and $\mathrm{Rev}(a',\mathbf{v})$ are different.

Consider for example a class of auctions defined on δ-discretized bid profiles where the revenue of every auction a on bid profile $\mathbf{v}$ is δ-discretized and remains the same even when all but a small set of k bidders reduce their bids. We denote the set of these bidders by $W_{a,\mathbf{v}} \subseteq [n]$, allowing for the dependence on the bid profile $\mathbf{v}$ and auction a. Examples of this include VCG auctions with bidder-specific reserves and the level auctions, where $W_{\mathbf{v},a}$ consists of the winner and the runner-up. Any two such different auctions must receive different revenues on a $2k$-sparse bid profile. This is due to the fact that if a and a' have different revenues on a bid profile $\mathbf{v}$, they also have different revenues on $\mathbf{v}'$, where $v'_i = v_i$ for $i \in W_{\mathbf{v},a} \cup W_{\mathbf{v},a'}$ and $v'_i = 0$ otherwise. Therefore, the set of $O\left(n^{2k}/\delta^{2k}\right)$, $2k$-sparse δ-discretized bid profiles distinguishes between any two auctions in this class and can serve as the columns of an implementable and δ-admissible matrix for it. This gives us an oracle-efficient algorithm for optimizing auctions with regret $O(n^{2k}\sqrt{T}/\delta^{2k+1})$.

When k is a small constant, such as in VCG with bidder-specific reserves and in level auctions, this simple construction of admissible and implementable matrices leads to a regret bound that is polynomial in the desired parameters. However, it is often possible to achieve better regret bounds by using more carefully designed translation matrices. We construct such matrices for each of the above auction classes. For VCG auctions with bidder-specific reserves and envy-free item-pricing auctions, we show how to implement an (obviously admissible) matrix $\mathbf{\Gamma}$, where each row corresponds, respectively, to the concatenated binary representation of bidder reserves or item prices. We show that, surprisingly, any perturbation that is a linear function of this binary representation can be simulated by a distribution of bidder valuations (see Figure 1 for an example). For the third class, level auctions, our challenge is to show that a clearly implementable matrix $\mathbf{\Gamma}$, with each column implemented by a single bid profile, is also admissible.

Table 1 summarizes regret bounds and runtimes of our oracle-efficient algorithms, assuming oracle calls run in time $O(1)$. Our algorithms perform a single oracle call per round, so T oracle calls in total. Our runtimes demonstrate an efficient reduction from online to offline learning.

While, in theory, the auction classes discussed in the table do not have worst-case polynomial-time algorithms for the offline problem, in practice, the offline problems are solved by a wide range of approaches, such as integer program solvers, which run fast on typical problem instances (e.g., see [51, 52]). Our approach can turn existing offline routines into online algorithms at almost no additional cost. When one wishes to leverage provably polynomial-time algorithms, even if their outputs are sub-optimal, our framework still applies as long as the algorithms return actions whose payoffs are within a constant factor of the best payoff. We discuss this approach briefly in Section 1.3 below and then demonstrate its applicability in Section 7, where we derive a polynomial-time algorithm for online welfare maximization in multi-unit auctions.

1.3 Extensions and Additional Applications

In Sections 4–7, we present several extensions and additional applications (see Table 2 for a summary):

Markovian Adversaries and Competing with the Optimal Auction (Section 4). Morgenstern and Roughgarden [43] show that level auctions can provide an arbitrarily accurate approximation to the overall optimal Myerson auction in the Bayesian single-item auction setting if the values of the bidders are drawn from independent distributions and i.i.d. over time. Therefore, if the environment in an online setting picks bidder valuations from independent distributions, standard online-to-batch reductions imply that the revenue of Generalized FTPL with the class of level auctions is close to the overall optimal (i.e., not just best-in-class) single-shot auction. We generalize this reasoning and show the same strong optimality guarantee when the valuations of bidders on each round are drawn from a fast-mixing Markov process that is independent across bidders but Markovian over rounds. For this setting, our results give an oracle-efficient algorithm with regret $O(n^{3/5}T^{9/10})$ to the overall optimal auction, rather than just best-in-class. This is the first result on competing with the Myerson optimal auction for non-i.i.d. distributions, as all prior work [13, 17, 43, 49] assumes i.i.d. samples.

Contextual Learning (Section 5). In this setting, on each round t the learner observes a context σ_t before choosing an action. For example, in online auction design, the context might represent demographic information about the set of bidders. The goal of the learner is to compete with the best *policy* in some fixed class, where each policy is a mapping from a context σ_t to an action. We propose a contextual extension of the translation matrix $\mathbf{\Gamma}$. Generalized FTPL can be applied using this extended translation matrix and provides sublinear regret bounds for both the case in which there is a small "separator" of the policy class and the transductive setting in which the set of all possible contexts is known ahead of time. Our results extend and generalize the results of Syrgkanis et al. [53] from contextual combinatorial optimization to any learning setting that admits an implementable and admissible translation matrix.

The contextual extension is particularly useful in online auction design, because it allows the learner to use any side information available about the bidders before they place their bids to guide the choice of auction. While the number of bidders might be too large to learn about them individually, the learner can utilize the side information to design a common treatment for bidders that are similar, that is, to *generalize across a population.*

Our performance guarantees for adaptive auction design, similar to much prior work, rely on the assumption that the bidders are either myopic or are different on each round. One criticism of this assumption is that such adaptive mechanisms might be manipulated by strategic bidders who distort their bids so as to gain in the future. The contextual learning algorithms mitigate this risk by pooling similar bidders, which reduces the probability that the exact same bidder will be overly influential in the choices of the algorithm.

Approximate Oracles and Approximate Regret (Section 6). For some problems there might not exist a sufficiently fast (e.g., polynomial-time or FPTAS) offline oracle with small additive error as required for Generalized FTPL. To make our results more applicable in practice, we extend them to handle oracles that are only required to return an action with performance that is within a constant multiplicative factor, $C \leq 1$, of that of the optimal action in the class. We consider two examples of such oracles: Relaxation-based Approximations [3] and Maximal-in-Range (MIR) algorithms [45]. Our results hold in both cases with a modified version of regret, called *C-regret*, in which the online algorithm competes with C times the payoff of the optimal action in hindsight.

Table 2. Additional results considered in Sections 4–7 and their significance. Above, m is the discretization level of the problems, n is the number of bidders, and T is the time horizon.

Problem Class	Regret	Section	Notes
Markovian, single item	$O(n^{3/5}T^{9/10})$	4.2	competes with Myerson's optimal auction
contextual online auction[a]	$O(\sqrt{T})$ or $O(T^{3/4})$	5	allows side information about bidders
welfare maximization, s-unit[b]	1/2-regret: $O(n^4\sqrt{T})$	7.1	fully (oracle-free) polynomial-time algorithm
bidding in SiSPAs, k items	$O(km\sqrt{T})$	7.2	solves an open problem [14]

[a] The two bounds are for the small separator setting and the transductive setting, respectively. Dependence on parameters other than T is omitted.

[b] The regime of interest in this problem is $s \gg n$. The extended abstract [20] contained a worse bound $O(n^6\sqrt{T})$.

Additional Applications (Section 7). Finally, we provide further applications of our work in the area of online combinatorial optimization with MIR approximate oracles, and in the area of no-regret learning for bid optimization in simultaneous second-price auctions.

- In the first application, we give a polynomial-time learning algorithm for online welfare maximization in multi-unit auctions that achieves 1/2-regret, by invoking the polynomial-time MIR approximation algorithm of Dobzinski and Nisan [19] as an offline oracle.
- In the second application, we solve an open problem raised in the recent work of Daskalakis and Syrgkanis [14], who offered efficient learning algorithms only for the weaker benchmark of no-envy learning, rather than no-regret learning, in simultaneous second-price auctions, and left open the question of oracle-efficient no-regret learning. We show that no-regret learning in simultaneous item auctions is efficiently achievable, assuming access to an optimal bidding oracle against a known distribution of opponents bids (equivalently, against a distribution of item prices).

2 GENERALIZED FTPL AND ORACLE-EFFICIENT ONLINE LEARNING

In this section, we introduce the Generalized Follow-the-Perturbed-Leader (Generalized FTPL) algorithm and describe the conditions under which it efficiently reduces online learning to offline optimization.

As described in Section 1.1, we consider the following online learning problem. On each round $t = 1, \ldots, T$, a learner chooses an action x_t from a finite set $\mathcal{X}$, and an adversary chooses an action y_t from a set $\mathcal{Y}$, which is not necessarily finite. The learner then observes y_t and receives a payoff $f(x_t, y_t) \in [0,1]$, where the function f is fixed and known to the learner. The goal of the learner is to obtain low expected regret with respect to the best action in hindsight, i.e., to minimize

$$\textsc{Regret} := \mathbb{E}\left[\max_{x\in\mathcal{X}} \sum_{t=1}^{T} f(x, y_t) - \sum_{t=1}^{T} f(x_t, y_t)\right],$$

where the expectation is over the randomness of the learner. An online algorithm is called a *no-regret algorithm* if its regret is sublinear in T, which means that its per-round regret goes to 0 as $T \to \infty$. To simplify exposition, we assume that the adversary is oblivious, i.e., that the sequence $y_1, \ldots, y_T$ is chosen up front without knowledge of the learner's realized actions. Our results generalize to adaptive adversaries using standard techniques [14, 34].

A natural first attempt at an online learning algorithm with oracle access would be one that simply invokes the oracle on the historical data at each round and plays the best action in hindsight. In a stochastic environment in which the adversary's actions are drawn i.i.d. from a fixed distribution, this *Follow-the-Leader* approach achieves a regret of $O(\sqrt{T \log |\mathcal{X}|})$. However, because the algorithm is deterministic, it performs poorly in adversarial environments [6].

To achieve sublinear regret, we use a common scheme, introduced by Kalai and Vempala [37], and optimize over a perturbed objective at each round. Indeed, our algorithm takes its name from Kalai and Vempala's *Follow-the-Perturbed-Leader* (FTPL) algorithm. Unlike FTPL, we do not generate a separate independent perturbation for each action, because this creates the two problems mentioned in Section 1.1. First, FTPL for unstructured payoffs requires creating $|\mathcal{X}|$ independent random variables, which is intractably large in many applications, including the auction design setting considered here. Second, FTPL yields optimization problems that require a stronger offline optimizer than assumed here. We overcome the first problem by working with perturbations that are not necessarily independent across different actions (prior instances of such an approach were only known for linear [37] and submodular [29] minimization). We address the second problem by implementing such perturbations with synthetic historical samples of adversary actions; this idea was introduced by Daskalakis and Syrgkanis [14], but they did not provide a method of randomly generating such samples in general learning settings. Thus, our work unifies and extends these previous lines of research.

We create shared randomness among actions in $\mathcal{X}$ by drawing a random vector $\boldsymbol{\alpha} \in \mathbb{R}^N$ of size N, with components α_j drawn independently from a dispersed distribution D. The payoff of each of the learner's actions is perturbed by a linear combination of these independent variables, as prescribed by a *perturbation translation matrix* $\boldsymbol{\Gamma}$ of size $|\mathcal{X}| \times N$, with entries in $[0,1]$. The rows of $\boldsymbol{\Gamma}$, denoted $\boldsymbol{\Gamma}_x$, describe the linear combination for each action x. That is, on each round t, the payoff of each learner action $x \in \mathcal{X}$ is perturbed by $\boldsymbol{\alpha} \cdot \boldsymbol{\Gamma}_x$, and our Generalized FTPL algorithm outputs an action x that approximately maximizes $\sum_{\tau=1}^{t-1} f(x, y_\tau) + \boldsymbol{\alpha} \cdot \boldsymbol{\Gamma}_x$. This procedure is fully described in Algorithm 1. (For non-oblivious adversaries, a fresh random vector $\boldsymbol{\alpha}$ is drawn in each round.)

Algorithm 1: Generalized FTPL

Input: matrix $\boldsymbol{\Gamma} \in [0,1]^{|\mathcal{X}| \times N}$, distribution D over $\mathbb{R}$, and optimization accuracy $\epsilon \geq 0$.
Draw $\alpha_j \sim D$ independently for $j = 1, \ldots, N$.
for $t = 1, \ldots, T$ **do**
  Choose any x_t such that for all $x \in \mathcal{X}$,

$$\sum_{\tau=1}^{t-1} f(x_t, y_\tau) + \boldsymbol{\alpha} \cdot \boldsymbol{\Gamma}_{x_t} \geq \sum_{\tau=1}^{t-1} f(x, y_\tau) + \boldsymbol{\alpha} \cdot \boldsymbol{\Gamma}_x - \epsilon.$$

  Observe y_t and receive payoff $f(x_t, y_t)$.
end for

In the remainder of this section, we analyze the properties of matrix $\boldsymbol{\Gamma}$ that guarantee that Generalized FTPL is no-regret and that its perturbations can be efficiently transformed into synthetic history. Together these properties give rise to efficient reductions of online learning to offline optimization.

2.1 Regret Analysis

To analyze Generalized FTPL, we first bound its regret by the sum of a *stability* term, a *perturbation* term, and an *error* term in the following lemma. While this approach is standard [37], we include a proof in Appendix B.1 for completeness.

Lemma 2.1 (ϵ-FTPL Lemma). *For Generalized FTPL, we have*

$$\textsc{Regret} \le \mathbb{E}\left[\sum_{t=1}^{T} f(x_{t+1}, y_t) - f(x_t, y_t)\right] + \mathbb{E}\left[\boldsymbol{\alpha} \cdot (\mathbf{\Gamma}_{x_1} - \mathbf{\Gamma}_{x^*})\right] + \epsilon(T+1), \tag{1}$$

where $x^* = \arg\max_{x \in \mathcal{X}} \sum_{t=1}^{T} f(x, y_t)$.

In this lemma, the first term measures the stability of the algorithm, i.e., how often the action changes from round to round. The second term measures the strength of the perturbation, that is, how much the perturbation amount differs between the best action and the initial action. The third term measures the aggregated approximation error in choosing x_t that only approximately optimizes $\sum_{\tau=1}^{t-1} f(x, y_\tau) + \boldsymbol{\alpha} \cdot \mathbf{\Gamma}_x$.

To bound the stability term, we require that the matrix $\mathbf{\Gamma}$ be *admissible* and the distribution D be *dispersed* in the following sense.

Definition 2.2 (δ-Admissible Translation Matrix). *A translation matrix* $\mathbf{\Gamma}$ *is* admissible *if its rows are distinct. It is* δ-admissible *if it is admissible and distinct elements within each column differ by at least* δ.

Definition 2.3 ((ρ, L)-Dispersed Distribution). *A distribution* D *on the real line is* (ρ, L)*-dispersed if for any interval of length* L*, the probability measure placed by* D *on this interval is at most* ρ.

In the next lemma, we bound the stability term in Equation (1) by showing that with high probability, for all rounds t, we have $x_{t+1} = x_t$. At a high level, since all rows of an admissible matrix $\mathbf{\Gamma}$ are distinct, it suffices to show that the probability that $\mathbf{\Gamma}_{x_{t+1}} \neq \mathbf{\Gamma}_{x_t}$ is small. We prove this for each coordinate $\Gamma_{x_{t+1}j}$ separately, by showing that it is only possible to have $\Gamma_{x_{t+1}j} \neq \Gamma_{x_t j}$ when the random variable α_j falls in a small interval, which happens with only small probability for a sufficiently dispersed distribution D.[4]

Lemma 2.4. *Consider Generalized FTPL with optimization accuracy* ϵ*, a* δ*-admissible matrix* $\mathbf{\Gamma}$ *with* N *columns, and a* $\left(\rho, \frac{1+2\epsilon}{\delta}\right)$*-dispersed distribution* D*. Then,* $\mathbb{E}\left[\sum_{t=1}^{T} f(x_{t+1}, y_t) - f(x_t, y_t)\right] \le 2TN\rho(1 + \delta^{-1})$.

Proof. Fix any $t \le T$. The bulk of the proof establishes that, with high probability, $\mathbf{\Gamma}_{x_{t+1}} = \mathbf{\Gamma}_{x_t}$, which by admissibility implies that $x_{t+1} = x_t$ and therefore $f(x_{t+1}, y_t) - f(x_t, y_t) = 0$.

Fix any $j \le N$. We first show that $\Gamma_{x_{t+1}j} = \Gamma_{x_t j}$ with high probability. Let V denote the set of values that appear in the j^{th} column of $\mathbf{\Gamma}$. By δ-admissibility, $|V| \le 1 + \delta^{-1}$. For any value $v \in V$, let x^v be any action that maximizes the perturbed cumulative payoff among those whose $\mathbf{\Gamma}$ entry in the j^{th} column equals v:

$$x^v \in \operatorname*{arg\,max}_{x \in \mathcal{X}:\, \Gamma_{xj} = v} \left[\sum_{\tau=1}^{t-1} f(x, y_\tau) + \boldsymbol{\alpha} \cdot \mathbf{\Gamma}_x\right] = \operatorname*{arg\,max}_{x \in \mathcal{X}:\, \Gamma_{xj} = v} \left[\sum_{\tau=1}^{t-1} f(x, y_\tau) + \boldsymbol{\alpha} \cdot \mathbf{\Gamma}_x - \alpha_j v\right].$$

For any $v, v' \in V$, define

$$\Delta_{vv'} = \left(\sum_{\tau=1}^{t-1} f(x^v, y_\tau) + \boldsymbol{\alpha} \cdot \mathbf{\Gamma}_{x^v} - \alpha_j v\right) - \left(\sum_{\tau=1}^{t-1} f(x^{v'}, y_\tau) + \boldsymbol{\alpha} \cdot \mathbf{\Gamma}_{x^{v'}} - \alpha_j v'\right).$$

Note that x^v and $\Delta_{vv'}$ are independent of α_j, as we removed the payoff perturbation corresponding to α_j.

[4]The proof of Lemma 2.4 implies a slightly tighter bound of $2TN\kappa\rho$, where κ is the maximum number of distinct elements in any column of $\mathbf{\Gamma}$. Note that δ-admissibility implies that $\kappa \le 1 + \delta^{-1}$.

If $\Gamma_{x_t j} = v$, then by the ϵ-optimality of x_t on the perturbed cumulative payoff, we have $\alpha_j(v' - v) - \epsilon \le \Delta_{vv'}$ for all $v' \in V$. Suppose $\Gamma_{x_{t+1} j} = v' \neq v$. Then by the optimality of $x^{v'}$ and the ϵ-optimality of x_{t+1},

$$\begin{aligned}\sum_{\tau=1}^{t-1} f(x^{v'}, y_\tau) + f(x_{t+1}, y_t) + \boldsymbol{\alpha} \cdot \boldsymbol{\Gamma}_{x^{v'}} &\ge \sum_{\tau=1}^{t-1} f(x_{t+1}, y_\tau) + f(x_{t+1}, y_t) + \boldsymbol{\alpha} \cdot \boldsymbol{\Gamma}_{x_{t+1}} \\ &\ge \sum_{\tau=1}^{t-1} f(x^{v}, y_\tau) + f(x^{v}, y_t) + \boldsymbol{\alpha} \cdot \boldsymbol{\Gamma}_{x^{v}} - \epsilon.\end{aligned}$$

Rearranging, we obtain for this same v' that

$$\Delta_{vv'} \le \alpha_j(v' - v) + f(x_{t+1}, y_t) - f(x^v, y_t) + \epsilon \le \alpha_j(v' - v) + 1 + \epsilon.$$

If $v' > v$, then

$$\alpha_j \ge \frac{\Delta_{vv'} - 1 - \epsilon}{v' - v} \ge \min_{\hat{v} \in V,\, \hat{v} > v} \frac{\Delta_{v\hat{v}} - 1 - \epsilon}{\hat{v} - v} \ ,$$

and so $\alpha_j(\overline{v} - v) + 1 + \epsilon \ge \Delta_{v\overline{v}}$ where $\overline{v}$ is the value of $\hat{v}$ minimizing the expression on the right. Thus, in this case we have $-\epsilon \le \Delta_{v\overline{v}} - \alpha_j(\overline{v} - v) \le 1 + \epsilon$. Similarly, if $v' < v$, then

$$\alpha_j \le \frac{\Delta_{vv'} - 1 - \epsilon}{v' - v} \le \max_{\hat{v} \in V,\, \hat{v} < v} \frac{\Delta_{v\hat{v}} - 1 - \epsilon}{\hat{v} - v} \ ,$$

and so $\alpha_j(\underline{v} - v) + 1 + \epsilon \ge \Delta_{v\underline{v}}$ where $\underline{v}$ is the value maximizing the expression on the right. In this case we have $-\epsilon \le \Delta_{v\underline{v}} - \alpha_j(\underline{v} - v) \le 1 + \epsilon$. Putting this all together, we have

$$\begin{aligned}&\Pr\left[\Gamma_{x_{t+1} j} \neq \Gamma_{x_t j} \,\middle|\, \alpha_k, k \neq j\right] \\ &\quad\le \Pr\left[\exists\, v \in V :\ -\epsilon \le \Delta_{v\overline{v}} - \alpha_j(\overline{v} - v) \le 1 + \epsilon \ \text{ or } \ -\epsilon \le \Delta_{v\underline{v}} - \alpha_j(\underline{v} - v) \le 1 + \epsilon \,\middle|\, \alpha_k, k \neq j\right] \\ &\quad\le \sum_{v \in V} \left(\Pr\left[\alpha_j \in \left[\tfrac{\Delta_{v\overline{v}} - 1 - \epsilon}{\overline{v} - v}, \tfrac{\Delta_{v\overline{v}} + \epsilon}{\overline{v} - v}\right] \,\middle|\, \alpha_k, k \neq j\right] + \Pr\left[\alpha_j \in \left[\tfrac{-\Delta_{v\underline{v}} - \epsilon}{v - \underline{v}}, \tfrac{-\Delta_{v\underline{v}} + 1 + \epsilon}{v - \underline{v}}\right] \,\middle|\, \alpha_k, k \neq j\right]\right) \\ &\quad\le 2|V|\rho \le 2\rho(1 + \delta^{-1}).\end{aligned}$$

The first inequality on the last line follows from the fact that $\overline{v} - v \ge \delta$ and $v - \underline{v} \ge \delta$, the fact that D is $\left(\rho, \frac{1+2\epsilon}{\delta}\right)$-dispersed, and a union bound. The final inequality follows because $|V| \le 1 + \delta^{-1}$ by δ-admissibility.

Since this bound does not depend on the values of α_k for $k \neq j$, we can remove the conditioning and bound $\Pr[\Gamma_{x_{t+1} j} \neq \Gamma_{x_t j}] \le 2\rho(1 + \delta^{-1})$. Taking a union bound over all $j \le N$, we then have that, by admissibility, $\Pr[x_{t+1} \neq x_t] = \Pr\left[\boldsymbol{\Gamma}_{x_{t+1}} \neq \boldsymbol{\Gamma}_{x_t}\right] \le 2N\rho(1 + \delta^{-1})$, which implies the result. □

To bound the regret, it remains to bound the perturbation term in Equation (1). This bound is specific to the distribution D. Many distribution families, including (discrete and continuous) uniform, Gaussian, Laplacian, and exponential can lead to a sublinear regret when the variance is set appropriately. In the majority of this work, we use D that is uniform on a subset of non-negative reals. This distribution works particularly well with our definition of oracle efficiency, which requires non-negative weights. However, when the payoff has a specific structure and the matrix $\boldsymbol{\Gamma}$ satisfies a stronger notion of implementability, we use a Gaussian distribution to achieve better regret bounds compared to the ones achievable by the uniform distribution (see Section 5.2). Here we present a concrete regret analysis for a uniform distribution:

Theorem 2.5. *Let $\mathbf{\Gamma}$ be a δ-admissible matrix with N columns and let D be the uniform distribution on $[0, 1/\eta]$ for $\eta = \delta/\sqrt{2T(1+2\epsilon)(1+\delta)}$. Then, the regret of Generalized FTPL can be bounded as*

$$\textsc{Regret} \leq \frac{N\sqrt{T}}{\delta} \cdot 2\sqrt{2(1+2\epsilon)(1+\delta)} \;+\; \epsilon(T+1) \;\;.$$

Proof. We use Lemmas 2.1 and 2.4 with an appropriate set of parameters. First note that for the uniform distribution on $[0, 1/\eta]$, the perturbation term in Lemma 2.1 is bounded by $\|\boldsymbol{\alpha}\|_1 \leq N/\eta$. Next note that the uniform distribution on $[0, 1/\eta]$ is $(\rho, (1+2\epsilon)\delta^{-1})$-dispersed with $\rho = \eta(1+2\epsilon)\delta^{-1}$. Therefore, the stability term in Lemma 2.1 can be bounded by Lemma 2.4 with this value of ρ. Combining the bounds on the perturbation and stability terms and using the value of η as stated in the theorem then yields

$$\begin{aligned}
\textsc{Regret} &\leq 2TN\rho\left(1+\frac{1}{\delta}\right) + \frac{N}{\eta} + \epsilon(T+1) \\
&= 2TN\eta(1+2\epsilon)\left(\frac{1}{\delta}\right)\left(1+\frac{1}{\delta}\right) + \frac{N}{\eta} + \epsilon(T+1) \\
&= 2TN\eta(1+2\epsilon)(1+\delta)\left(\frac{1}{\delta}\right)^2 + \frac{N}{\eta} + \epsilon(T+1) \\
&= \frac{N\sqrt{T}}{\delta} \cdot 2\sqrt{2(1+2\epsilon)(1+\delta)} + \epsilon(T+1) \;\;. \qquad \square
\end{aligned}$$

Throughout the paper we consider $\epsilon = 1/\sqrt{T}$, which is the weakest setting with respect to T that does not negatively impact the regret rate regardless of other problem-dependent constants such as N and δ:

Corollary 2.6. *Let $\mathbf{\Gamma}$ be a δ-admissible matrix with N columns, let D be the uniform distribution on $[0, 1/\eta]$ for $\eta = \delta/\sqrt{2T(1+2T^{-1/2})(1+\delta)}$, and let $\epsilon = 1/\sqrt{T}$. Then the regret of Generalized FTPL is $O(N\sqrt{T}/\delta)$.*

We note that for any function $f : \mathcal{X} \times \mathcal{Y} \to [0,1]$ there is a 1-admissible matrix $\mathbf{\Gamma}$ with $N = \lceil \log_2 |\mathcal{X}| \rceil$ columns whose rows contain binary representations of numbers $1, \ldots, |\mathcal{X}|$. More generally, for any $\delta \in (0,1]$, there is a δ-admissible matrix $\mathbf{\Gamma}$ with $N = \lceil \log_{\lfloor 1+\delta^{-1} \rfloor} |\mathcal{X}| \rceil$ columns, whose rows contain scaled representations of numbers $1, \ldots, |\mathcal{X}|$ in base $\lfloor 1 + \delta^{-1} \rfloor$. Corollary 2.6 then implies the regret bound $O\Big((\log_{\lfloor 1+\delta^{-1} \rfloor} |\mathcal{X}|)\sqrt{T}/\delta\Big)$, which is tightest for $\delta = 1$, when it becomes $O\Big((\log |\mathcal{X}|)\sqrt{T}\Big)$. This is worse than the regret $O(\sqrt{T \log |\mathcal{X}|})$ achieved by the multiplicative weights algorithms [24, 38].

However, the multiplicative weights algorithms typically require running times that are polynomial in $|\mathcal{X}|$, whereas our algorithm can be exponentially faster assuming access to efficient optimization oracles. We therefore next turn our attention to the analysis of the running time of our algorithm.

2.2 Oracle-Efficient Online Learning

We now define the offline oracle and oracle-efficient online learning more formally. Our oracles are defined for real-weighted datasets, but can be easily implemented by integer-weighted oracles (see the reduction in Appendix F.2). Since many natural offline oracles are iterative optimization algorithms, which are only guaranteed to return an approximate solution in finite time, our definition assumes that the oracle takes the desired precision ϵ as an input. For ease of exposition, we assume that all numerical computations, even those involving real numbers, take $O(1)$ time. We discuss this point in more detail in Appendix F.

Definition 2.7 (Offline Oracle). *An* offline oracle Opt *is any algorithm that receives as input a weighted set of adversary actions $S = \{(w_\ell, y_\ell)\}_{\ell \in \mathcal{L}}$ with $w_\ell \in \mathbb{R}^+$, $y_\ell \in \mathcal{Y}$ and a desired precision ϵ, and returns an action $\hat{x} = \textsc{Opt}(S, \epsilon)$ such that*

$$\sum_{(w,y)\in S} w f(\hat{x}, y) \geq \max_{x \in \mathcal{X}} \sum_{(w,y)\in S} w f(x, y) - \epsilon.$$

Definition 2.8 (Oracle Efficiency). *We say that an online algorithm is* oracle-efficient *with per-round complexity $g(\ldots)$ if its per-round running time is $O(g(\ldots))$ with oracle calls counting $O(1)$. The notation $g(\ldots)$ refers to the fact that g may be a function of problem-specific parameters, including T.*

In the definition of oracle efficiency, we intentionally ignore the running time of the offline oracle, which will depend on the problem size as well as the desired precision ϵ, which we set equal to $1/\sqrt{T}$ throughout the paper. Later in this section (see Corollary 2.6), we revisit the offline-oracle running time and show that polynomial-time oracles, that is those where $\textsc{Opt}\left(S, \frac{1}{\sqrt{T}}\right)$ runs in time $\mathrm{poly}(|S|, T)$, give rise to polynomial-time Generalized FTPL.

We next define a property of a translation matrix $\mathbf{\Gamma}$ which allows us to transform the perturbed objective into a dataset, thus achieving oracle efficiency of Generalized FTPL:

Definition 2.9. *A matrix $\mathbf{\Gamma}$ is* implementable *with complexity M if for each $j \in [N]$ there exists a weighted dataset S_j, with $|S_j| \leq M$, such that*

$$\text{for all } x, x' \in \mathcal{X}: \quad \Gamma_{xj} - \Gamma_{x'j} = \sum_{(w,y)\in S_j} w\Big(f(x,y) - f(x',y)\Big).$$

In other words, for each $j \in [N]$ there is $c_j \in \mathbb{R}$, such that for all $x \in \mathcal{X}$, $\Gamma_{xj} = c_j + \sum_{(w,y)\in S_j} w f(x,y)$. We say that weighted datasets S_j, $j \in [N]$, implement $\mathbf{\Gamma}$ with complexity M.

One simple but useful example of implementability is when each column j of $\mathbf{\Gamma}$ specifies the payoffs of learner's actions under a particular adversary action $y_j \in \mathcal{Y}$, i.e., $\Gamma_{xj} = f(x, y_j)$. In this case, $S_j = \{(1, y_j)\}$.

Using an implementable $\mathbf{\Gamma}$ gives rise to an oracle-efficient variant of the Generalized FTPL, provided in Algorithm 2, in which we explicitly set $\epsilon = 1/\sqrt{T}$. Theorem 2.10 shows that the output of this algorithm is equivalent to the output of Generalized FTPL and therefore the same regret guarantees hold. Note the assumption that the perturbations α_j are non-negative. The algorithm can be extended to negative perturbations when both $\mathbf{\Gamma}$ and $-\mathbf{\Gamma}$ are implementable. (See Section 5 for details.)

Theorem 2.10. *If $\mathbf{\Gamma}$ is implementable with complexity M, then Algorithm 2 is an oracle-efficient implementation of Algorithm 1 with $\epsilon = 1/\sqrt{T}$ and has per-round complexity $O\Big(T + NM\Big)$.*

Proof. To show that the Oracle-Based FTPL procedure (Algorithm 2) implements Generalized FTPL (Algorithm 1) with $\epsilon = 1/\sqrt{T}$, it suffices to show that at each round t, for $S = \Big\{(1, y_1), \ldots, (1, y_{t-1})\Big\} \cup \bigcup_{j \leq N} \alpha_j S_j$, and for any x,

$$\sum_{\tau=1}^{t-1} f(x, y_\tau) + \boldsymbol{\alpha} \cdot \mathbf{\Gamma}_x \geq \max_{x' \in \mathcal{X}} \left[\sum_{\tau=1}^{t-1} f(x', y_\tau) + \boldsymbol{\alpha} \cdot \mathbf{\Gamma}_{x'}\right] - \epsilon$$

$$\iff \sum_{(w,y)\in S} w f(x,y) \geq \max_{x' \in \mathcal{X}} \sum_{(w,y)\in S} w f(x',y) - \epsilon. \tag{2}$$

Algorithm 2: Oracle-Based Generalized FTPL

Input: datasets S_j, $j \in [N]$, that implement a matrix $\mathbf{\Gamma} \in [0,1]^{|\mathcal{X}|\times N}$,
distribution D over $\mathbb{R}^+$,
an offline oracle Opt.
Draw $\alpha_j \sim D$ independently for $j = 1,\dots,N$.
For all j, let $\alpha_j S_j$ denote the scaled version of S_j, i.e., $\alpha_j S_j := \{(\alpha_j w, y) : (w,y) \in S_j\}$.
for $t = 1,\dots,T$ **do**
Set $S = \left\{(1,y_1),\dots,(1,y_{t-1})\right\} \cup \bigcup_{j\le N} \alpha_j S_j$.
Play $x_t = \textsc{Opt}\left(S, \frac{1}{\sqrt{T}}\right)$.
Observe y_t and receive payoff $f(x_t,y_t)$.
end for

Note that if the above equivalence holds, then in each round, the set of actions that $\textsc{Opt}(S,\epsilon)$ can return legally, i.e., actions whose payoffs are an approximation of the optimal payoff, is exactly the same as the set of actions that the offline optimization step of Algorithm 1 can legally play. Therefore, the theorem is proved if Equation (2) holds.

Let us show that Equation (2) is indeed true. Consider any $x,x' \in \mathcal{X}$. Then, from the definition of S and by implementability,

$$\begin{aligned}\sum_{(w,y)\in S} w f(x,y) - \sum_{(w,y)\in S} w f(x',y) &= \sum_{\tau=1}^{t-1}\left[f(x,y_\tau) - f(x',y_\tau)\right] + \sum_{j\in[N]}\alpha_j \sum_{(w,y)\in S_j} w\left(f(x,y) - f(x',y)\right)\\ &= \sum_{\tau=1}^{t-1}\left[f(x,y_\tau) - f(x',y_\tau)\right] + \sum_{j\in[N]}\alpha_j(\Gamma_{xj} - \Gamma_{x'j})\\ &= \left(\sum_{\tau=1}^{t-1} f(x,y_\tau) + \boldsymbol{\alpha}\cdot\mathbf{\Gamma}_x\right) - \left(\sum_{\tau=1}^{t-1} f(x',y_\tau) + \boldsymbol{\alpha}\cdot\mathbf{\Gamma}_{x'}\right),\end{aligned}$$

which immediately yields Equation (2).

Also, by implementability, the running time to construct the set S is at most $O(T+NM)$. Since there is only one oracle call per round, we get the per-round complexity of $O(T+NM)$. □

As an immediate corollary, we obtain that the existence of a polynomial-time offline oracle implies the existence of polynomial-time online learner with regret $O(\sqrt{T})$, whenever we have access to an implementable and admissible matrix.

Corollary 2.11. *Assume that $\mathbf{\Gamma} \in [0,1]^{|\mathcal{X}|\times N}$ is implementable with complexity M and δ-admissible, and there exists an approximate offline oracle $\textsc{Opt}\left(S, \frac{1}{\sqrt{T}}\right)$ which runs in time $\mathrm{poly}(|S|,T)$. Then Algorithm 2 with distribution D as defined in Theorem 2.5 runs in time $\mathrm{poly}(N,M,T)$ and achieves regret $O(N\sqrt{T}/\delta)$.*

Alternative Notions of Oracles. Throughout this paper we work with offline optimization oracles that take data with arbitrary non-negative real weights as their input. In Corollary 2.11, we then consider oracles of this form that run in polynomial time. Other notions of oracles may be natural in various applications. For instance, instead of real-weighted, one can consider integer-weighted oracles, and instead of polynomial-time, one can consider pseudo-polynomial oracles. We discuss these alternative notions in Appendix F. We show that integer-weighted oracles can be used to implement approximately optimal real-weighted oracles, so all of our results immediately extend

to integer-weighted oracles. For pseudo-polynomial oracles, the running time of the algorithm depends on the pseudo-complexity of the datasets that implement $\mathbf{\Gamma}$ (pseudo-complexity is defined in Appendix F.1). In that case, for instance, the magnitude of the weights implementing the matrix $\mathbf{\Gamma}$ affects the final running time of the learning algorithm. In Appendix F.4 we show that the pseudo-complexities of matrices $\mathbf{\Gamma}$ constructed in the next section are polynomial in the parameters of interest, so in those cases even pseudo-polynomial offline oracles give rise to polynomial-time no-regret algorithms.

3 ONLINE AUCTION DESIGN

In this section, we apply the general techniques developed in Section 2 to obtain oracle-efficient no-regret algorithms for several common auction classes.

Consider a mechanism-design setting in which a seller wants to allocate $k \geq 1$ heterogeneous resources to a set of n bidders. The allocation to a bidder i is a subset of $\{1,\dots,k\}$, which we represent as a vector in $\{0,1\}^k$, and the seller has some feasibility constraints on the allocations across bidders. Each bidder $i \in [n]$ has a combinatorial valuation function $v_i \in \mathcal{V}$, where $\mathcal{V} \subseteq \left(\{0,1\}^k \to [0,1]\right)$. We use $\mathbf{v} \in \mathcal{V}^n$ to denote the vector of valuation functions across all bidders. A special case of the setting is that of multi-item auctions for k heterogeneous items, where each resource is an item and the feasibility constraint simply states that no item is allocated to more than one bidder. Another special case is that of *single-parameter (service-based) environments*, which we describe in more detail in Section 3.1.

An auction a takes as input a *bid profile* consisting of reported valuations for each bidder, and returns both the allocation for each bidder i and the price that he is charged. In this work, we only consider *truthful auctions*, where each bidder maximizes his utility by reporting his true valuation, irrespective of what other bidders report. We therefore make the assumption that each bidder reports v_i as their bid and refer to $\mathbf{v}$ not only as the valuation profile, but also as the bid profile throughout the rest of this section. The allocation that the bidder i receives is denoted $\mathbf{q}_i(\mathbf{v}) \in \{0,1\}^k$ and the price that he is charged is $p_i(\mathbf{v})$; we allow sets $\mathbf{q}_i(\mathbf{v})$ to overlap across bidders, and drop the argument $\mathbf{v}$ when it is clear from the context. We consider bidders with quasilinear utilities: the utility of bidder i is $v_i(\mathbf{q}_i(\mathbf{v})) - p_i(\mathbf{v})$. For an auction a with price function $\mathbf{p}(\cdot)$, we denote by $\mathrm{Rev}(a,\mathbf{v})$ the *revenue of the auction* for bid profile $\mathbf{v}$, i.e., $\mathrm{Rev}(a,\mathbf{v}) = \sum_{i\in[n]} p_i(\mathbf{v})$.

Fixing a class of truthful auctions $\mathcal{A}$ and a set of possible valuations $\mathcal{V}$, we consider the problem in which on each round $t = 1,\dots,T$, a learner chooses an auction $a_t \in \mathcal{A}$ while an adversary chooses a bid profile $\mathbf{v}_t \in \mathcal{V}^n$. The learner then observes $\mathbf{v}_t$ and receives revenue $\mathrm{Rev}(a_t,\mathbf{v}_t)$. The goal of the learner is to obtain low expected regret with respect to the best auction from $\mathcal{A}$ in hindsight. That is, we would like to guarantee that

$$\textsc{Regret} := \mathbb{E}\left[\max_{a\in\mathcal{A}} \sum_{t=1}^{T} \mathrm{Rev}(a,\mathbf{v}_t) - \sum_{t=1}^{T} \mathrm{Rev}(a_t,\mathbf{v}_t)\right] \leq o(T)\mathrm{poly}(n,k).$$

We require our online algorithm to be oracle-efficient, assuming access to an ϵ-optimal offline optimization oracle that takes as input a weighted set of bid profiles, $S = \{(w_\ell,\mathbf{v}_\ell)\}_{\ell\in\mathcal{L}}$, and returns an auction that achieves an approximately optimal revenue on S, i.e., a revenue at least $\max_{a\in\mathcal{A}} \sum_{(w,\mathbf{v})\in S} w\mathrm{Rev}(a,\mathbf{v}) - \epsilon$. Throughout the section, we assume that there exists such an oracle for $\epsilon = 1/\sqrt{T}$, as needed in Algorithm 2.

Using the language of oracle-based online learning developed in Section 2, the learner's action corresponds to the choice of auction, the adversary's action corresponds to the choice of bid profile, the payoff of the learner corresponds to the revenue generated by the auction, and we assume access

to an offline optimization oracle Opt. These correspondences are summarized in the following table.

Auction Setting	**Oracle-Based Learning Equivalent**
Auctions $a_t \in \mathcal{A}$	Learner actions $x_t \in \mathcal{X}$
Bid/valuation profiles $\mathbf{v}_t \in \mathcal{V}^n$	Adversary actions $y_t \in \mathcal{Y}$
Revenue function Rev	Payoff function f

For several of the auction classes we consider, such as multi-item or multi-unit auctions, the revenue of an auction on a bid profile is in range $[0,R]$ for $R > 1$. In order to use the results of Section 2, we implicitly re-scale all the revenue functions by dividing them by R before applying Theorem 2.5. Note that, since $\mathbf{\Gamma}$ does not change, the admissibility condition keeps the regret of the normalized problem at $O(N\sqrt{T}/\delta)$, according to Theorem 2.5. We then scale up to get a regret bound that is R times the regret for the normalized problem, i.e., $O(RN\sqrt{T}/\delta)$. This re-scaling does not increase the runtime, because the complexity of implementing $\mathbf{\Gamma}$ is unchanged, only the weights appearing in sets S_j are scaled up by a factor of R, and we assume that all numerical computations take $O(1)$ time. Refer to Appendix F for a note on numerical computations and the mild change in runtime when numerical computations do not take $O(1)$ time.

We now derive results for three auction classes: VCG auctions with bidder-specific reserves, envy-free item-pricing auctions, and level auctions. Each auction class is formally defined in its respective subsection.

3.1 VCG with Bidder-Specific Reserves

In this section, we consider a standard class of auctions, VCG auctions with bidder-specific reserve prices, which we define more formally below and denote by $\mathcal{I}$. These auctions are known to approximately maximize the revenue when bidder valuations are drawn from independent (but not necessarily identical) distributions [28]. Recently, Roughgarden and Wang [50] considered online learning for this class and provided a computationally efficient algorithm whose total revenue is at least 1/2 of the best revenue among auctions in $\mathcal{I}$, minus a term that is $o(T)$. We apply the techniques from Section 2 to generate an oracle-efficient online algorithm with low *additive* regret with respect to the optimal auction in the class $\mathcal{I}$, without any loss in multiplicative factors.

We go beyond single-item auctions and consider general *single-parameter* environments. In these environments, each bidder has one piece of private valuation for receiving a *service*, i.e., being included in the set of winning bidders. We allow for some combinations of bidders to be *served* simultaneously, and let $\mathcal{S} \subseteq 2^{[n]}$ be the family of feasible sets, i.e., sets of bidders that can be served simultaneously; with some abuse of notation we write $\mathbf{q} \in \mathcal{S}$, to mean that the set represented by the binary allocation vector $\mathbf{q}$ is in $\mathcal{S}$. We assume that any bidder is allowed to be the sole bidder served, i.e., that $\{i\} \in \mathcal{S}$ for all i, and that it is also allowed that no bidder be served, i.e., $\emptyset \in \mathcal{S}$.[5] Examples of such environments include single-item single-unit auctions (for which $\mathcal{S}$ contains only singletons and the empty set), single-item s-unit auctions (for which $\mathcal{S}$ contains any subset of size at most s), and combinatorial auctions with single-minded bidders. In the last case, we begin with some set of original items, define the service as receiving the desired bundle of items, and let $\mathcal{S}$ contain any subset of bidders seeking disjoint sets of items.

In a basic VCG auction, an allocation $\mathbf{q}^* \in \mathcal{S}$ is chosen to maximize social welfare, that is, maximize $\sum_{i=1}^n v_i q_i^*$, where we slightly simplify notation and use $v_i \in [0,1]$ to denote the valuation of bidder i for being served. Each bidder who is served is then charged the externality he imposes

[5] A more common and stronger assumption used in previous work [28, 50] is that $\mathcal{S}$ is a downward-closed matroid.

on others, $p_i(\mathbf{v}) = \max_{\mathbf{q}\in\mathcal{S}} \sum_{i'\neq i} v_{i'} q_{i'} - \sum_{i'\neq i} v_{i'} q^*_{i'}$, which can be shown to equal the minimum bid at which he would be served. Such auctions are known to be truthful. The most common example is the second-price auction for the single-item single-unit case in which the bidder with the highest bid receives the item and pays the second highest bid. VCG auctions with reserves, which maintain the property of truthfulness, are defined as follows.

Definition 3.1 (VCG auctions with bidder-specific reserves). *A VCG auction with bidder-specific reserves is specified by a vector* $\mathbf{r}$ *of reserve prices for each bidder. As a first step, all bidders whose bids are below their reserves (that is, bidders* i *for which* $v_i < r_i$*) are removed from the auction. If no bidders remain, no item is allocated. Otherwise, the basic VCG auction is run on the remaining bidders to determine the allocation. Each bidder who is served is charged the larger of his reserve and his VCG payment.*

Fixing the set $\mathcal{S}$ of feasible allocations, we denote by $\mathcal{I}$ the class of all VCG auctions with bidder-specific reserves. With a slight abuse of notation we write $\mathbf{r} \in \mathcal{I}$ to denote the auction with reserve prices $\mathbf{r}$. To apply the results from Section 2, which require a finite action set for the learner, we limit attention to the finite set of auctions $\mathcal{I}_m \subseteq \mathcal{I}$ consisting of those auctions in which the reserve price for each bidder is a strictly positive integer multiple of $1/m$, i.e., those where $r_i \in \{1/m, \ldots, m/m\}$ for all i. We will show for some common choices of $\mathcal{S}$ that the best auction in this class yields almost as high a revenue as the best auction in $\mathcal{I}$.

We next show how to design a matrix $\mathbf{\Gamma}$ for $\mathcal{I}_m$ that is admissible and implementable. As a warmup, suppose we use the $|\mathcal{I}_m| \times n$ matrix $\mathbf{\Gamma}$ with entries $\Gamma_{\mathbf{r},i} = \mathrm{Rev}(\mathbf{r}, \mathbf{e}_i)$. That is, the i^{th} column of $\mathbf{\Gamma}$ corresponds to the revenue of each auction on a bid profile in which bidder i has valuation 1 and all others have valuation 0. By definition, $\mathbf{\Gamma}$ is implementable with complexity 1 using $S_j = \{(1, \mathbf{e}_j)\}$ for each $j \in [n]$. Moreover, $\mathrm{Rev}(\mathbf{r}, \mathbf{e}_i) = r_i$ so any two rows of $\mathbf{\Gamma}$ are different and $\mathbf{\Gamma}$ is thus $1/m$-admissible. By Theorems 2.5 and 2.10, we obtain an oracle-efficient implementation of the Generalized FTPL with regret $O(nm\sqrt{T})$.

To improve this regret bound and obtain a regret that is polynomial in $\log m$ rather than m, we carefully construct another translation matrix that is implementable using a more complex dataset of adversarial actions. The translation matrix we design is quite intuitive. The row corresponding to an auction $\mathbf{r}$ contains a binary representation of its reserve prices. In this case, proving admissibility of the matrix is simple. The challenge is to show that this simple translation matrix is implementable using a dataset of adversarial actions.

Construction of $\mathbf{\Gamma}$*:* Let $\mathbf{\Gamma}^{\mathrm{VCG}}$ be an $|\mathcal{I}_m| \times (n\lceil \log m \rceil)$ binary matrix, where the i^{th} collection of $\lceil \log m \rceil$ columns contains the binary encodings of the auctions' reserve prices for bidder i. More formally, for any $i \leq n$ and a bit position $\beta \leq \lceil \log m \rceil$, let $j = (i-1)\lceil \log m \rceil + \beta$ and set $\mathbf{\Gamma}^{\mathrm{VCG}}_{\mathbf{r},j}$ to be the β^{th} bit of mr_i.

In Lemma 3.3, we prove that $\mathbf{\Gamma}^{\mathrm{VCG}}$ is implementable and admissible. But first, let us illustrate the main ideas of this proof through a simple example.

Example 3.2. *Consider* $\mathbf{\Gamma}^{\mathrm{VCG}}$ *for* $n = 2$ *bidders and* $m = 3$ *discretization levels, as demonstrated in Figure 1. As an example, we show how one can go about implementing columns* 1 *and* 4 *of* $\mathbf{\Gamma}^{\mathrm{VCG}}$*.*

Consider the first column of $\mathbf{\Gamma}^{\mathrm{VCG}}$*. It corresponds to the most significant bit of* r_1*. To implement this column, we need to find a weighted set of bid profiles that generate revenues with the same differences as those between the column entries. We consider bid profiles* $\mathbf{v}_h = (h/3, 0)$ *for* $h \in \{1, 2, 3\}$*, with the revenue* $\mathrm{Rev}(\mathbf{r}, \mathbf{v}_h) = r_1 \mathbf{1}_{(h/3 \geq r_1)}$*. To obtain the weights* w_h *for each* $\mathbf{v}_h$ *it is necessary (and sufficient) to match differences between entries corresponding to reserve prices with* $r_1 = \frac{1}{3}$ *vs* $\frac{2}{3}$*, and* $r_1 = \frac{2}{3}$ *vs* $\frac{3}{3}$

	Binary encoding			
Auction $\mathbf{r}$	r_1		r_2	
$(1/3, 1/3)$	0	1	0	1
$(1/3, 2/3)$	0	1	1	0
$(1/3, 3/3)$	0	1	1	1
$(2/3, 1/3)$	1	0	0	1
$(2/3, 2/3)$	1	0	1	0
$(2/3, 3/3)$	1	0	1	1
$(3/3, 1/3)$	1	1	0	1
$(3/3, 2/3)$	1	1	1	0
$(3/3, 3/3)$	1	1	1	1

$\Delta = -1$
$\Delta = 0$
$\Delta' = 1$
$\Delta' = -1$

Fig. 1. $\mathbf{\Gamma}^{\text{VCG}}$ for $n = 2$ bidders and discretization $m = 3$.

(denoted by Δ in Figure 1), corresponding to the following equations:

$$\frac{1}{3}(w_1 + w_2 + w_3) - \frac{2}{3}(w_2 + w_3) = -1,$$
$$\frac{2}{3}(w_2 + w_3) - \frac{3}{3}(w_3) = 0,$$

where the left-hand sides are the differences in the revenues and the right-hand sides are the differences Δ between the corresponding column entries. This is an underdetermined system of linear equations, for which we need to find a non-negative solution. One such solution is provided by the weights $w_1 = 3$, $w_2 = 2$ and $w_3 = 4$. Thus, the weighted set $S_1 = \{(3, \mathbf{v}_1), (2, \mathbf{v}_2), (4, \mathbf{v}_3)\}$ implements the first column. Similarly, for implementing the fourth column, we consider bid profiles $\mathbf{v}'_h = (0, h/3)$ for $h \in \{1, 2, 3\}$ and equations dictated by the differences Δ'. One can verify that $S_4 = \{(6, \mathbf{v}'_1), (0, \mathbf{v}'_2), (3, \mathbf{v}'_3)\}$ implements this column.

More generally, the proof of Lemma 3.3 shows that $\mathbf{\Gamma}^{VCG}$ is implementable by showing that any differences in values in one column that solely depend on a single bidder's reserve price lead to a feasible system of linear equations.

Lemma 3.3. *$\mathbf{\Gamma}^{\text{VCG}}$ is 1-admissible and implementable with complexity m.*

Proof. In the interest of readability, we drop the superscript and write $\mathbf{\Gamma}$ for $\mathbf{\Gamma}^{\text{VCG}}$ in this proof.

For any $\mathbf{r}$, row $\mathbf{\Gamma}_{\mathbf{r}}$ corresponds to the binary encoding of $r_1, \ldots, r_n$. Therefore, for any two different auctions $\mathbf{r} \neq \mathbf{r}'$, we have $\mathbf{\Gamma}_{\mathbf{r}} \neq \mathbf{\Gamma}_{\mathbf{r}'}$. Since $\mathbf{\Gamma}$ is a binary matrix, this implies that $\mathbf{\Gamma}$ is 1-admissible.

Next, we will construct the sets S_j that implement each column $j \leq n\lceil \log m \rceil$. Pick $i \leq n$ and $\beta \leq \lceil \log m \rceil$, and the associated column index j. The set S_j includes exactly the m profiles in which only the bidder i has non-zero valuation, denoted as $\mathbf{v}_h := (h/m)\mathbf{e}_i$ for $h \leq m$. To determine their weights w_h, we use the definition of implementability. In particular, the weights must satisfy:

$$\forall\, \mathbf{r}, \mathbf{r}' \in \mathcal{I}_m, \qquad \Gamma_{\mathbf{r},j} - \Gamma_{\mathbf{r}',j} = \sum_{h \leq m} w_h \Big(\text{Rev}(\mathbf{r}, \mathbf{v}_h) - \text{Rev}(\mathbf{r}', \mathbf{v}_h) \Big).$$

In the above equation, $\Gamma_{\mathbf{r},j}$ and $\Gamma_{\mathbf{r}',j}$ encode the β^{th} bit of r_i and r'_i, respectively, so the left-hand side is independent of the reserve prices for bidders $i' \neq i$. Moreover, $\text{Rev}(\mathbf{r}, \mathbf{v}_h) = r_i \mathbf{1}_{(h \geq m r_i)}$, so the right-hand side of the above equation is also independent of the reserve prices for bidders $i' \neq i$. Let z_β be the β^{th} bit of integer z. That is, $\Gamma_{\mathbf{r},j} = (m r_i)_\beta$. Substituting $z = m r_i$ and $z' = m r'_i$, the above

equation can be reformulated as[6]

$$\forall z, z' \in \{1, \dots, m\}, \quad \left(z_\beta - z'_\beta\right) = \sum_{h \le m} w_h \left(\frac{z}{m}\mathbf{1}_{(h \ge z)} - \frac{z'}{m}\mathbf{1}_{(h \ge z')}\right). \tag{3}$$

We recursively derive the weights w_h, and show that they are non-negative and satisfy Equation (3). To begin, let

$$w_m = \max\left\{0,\ \max_z\left[m\left(z_\beta - (z-1)_\beta\right)\right]\right\},$$

and for all $z = m, m-1, \dots, 2$, define

$$w_{z-1} = \frac{1}{z-1}\left(\sum_{h=z}^{m} w_h - m\left(z_\beta - (z-1)_\beta\right)\right).$$

Next, we show by induction that $w_h \ge 0$ for all h. For the base case of $h = m$, by definition $w_m \ge 0$. Now, assume that for all $h \ge z$, $w_h \ge 0$. Then

$$w_{z-1} \ge \frac{1}{z-1}\left(w_m - m\left(z_\beta - (z-1)_\beta\right)\right) \ge 0.$$

Therefore all weights are non-negative. Furthermore, by rearranging the definition of w_{z-1}, we have

$$\begin{aligned}\left(z_\beta - (z-1)_\beta\right) &= \frac{1}{m}\left(\sum_{h=z}^{m} w_h - (z-1)w_{z-1}\right) = \frac{1}{m}\left(z\sum_{h=z}^{m} w_h - (z-1)\sum_{h=z-1}^{m} w_h\right) \\ &= \sum_{h \le m} w_h \left(\frac{z}{m}\mathbf{1}_{(h \ge z)} - \frac{z-1}{m}\mathbf{1}_{(h \ge z-1)}\right),\end{aligned}$$

where in the second equality we simply added and subtracted the term $(z-1)\sum_{h=z}^{m} w_h$ and in the last equality, we grouped together common terms.

Equation (3) is proved for a particular pair $z > z'$ by summing the above expression for $\left(\zeta_\beta - (\zeta - 1)_\beta\right)$ over all $\zeta \in (z', z]$ and canceling telescoping terms, and if $z = z'$, the statement holds regardless of the weights chosen.

This shows that $\mathbf{\Gamma}$ is implementable. Note that the cardinality of each S_j is m, so $\mathbf{\Gamma}$ is implementable with complexity m. □

The next theorem follows immediately from Lemma 3.3, Theorems 2.5 and 2.10, and the fact that the maximum revenue is at most R. Note that R is bounded by the number of bidders that can be served simultaneously, which is at most n.

Theorem 3.4. *Consider the online auction design problem for the class of VCG auctions with bidder-specific reserves, $\mathcal{I}_m$. Let $R = \max_{\mathbf{r},\mathbf{v}} \mathrm{Rev}(\mathbf{r}, \mathbf{v})$ and let D be the uniform distribution as described in Theorem 2.5. Then, the Oracle-Based Generalized FTPL algorithm with D and datasets that implement $\mathbf{\Gamma}^{\mathrm{VCG}}$ is oracle-efficient with per-round complexity $O(T + nm \log m)$ and has regret*

$$\mathbb{E}\left[\max_{\mathbf{r} \in \mathcal{I}_m} \sum_{t=1}^{T} \mathrm{Rev}(\mathbf{r}, \mathbf{v}_t) - \sum_{t=1}^{T} \mathrm{Rev}(\mathbf{r}_t, \mathbf{v}_t)\right] \le O(nR\sqrt{T}\log m).$$

Now we return to the infinite class $\mathcal{I}$ of all VCG auctions with reserve prices $r_i \in [0, 1]$. We show that $\mathcal{I}_m$ is a finite "cover" for this class when the family of feasible sets $\mathcal{S}$ consists of all subsets of size at most s, corresponding to single-item single-unit auctions (when $s = 1$) or more general single-item s-unit auctions. In such auctions, the items are allocated to the s highest bids that are

[6]Not including the reserve 0 is a crucial technical point for the proof of implementability.

above their reserve, and each winner pays the larger of its reserve price and the $s+1^{\text{st}}$ highest bid that had cleared its respective reserve price. We assume that the ties are resolved in favor of bidders with a lower index. We prove in Appendix C.1 that for these auctions, the optimal revenue of $\mathcal{I}_m$ compared with that of $\mathcal{I}$ can decrease by at most s/m at each round. That is,

$$\max_{\mathbf{r}\in\mathcal{I}} \sum_{t=1}^{T} \text{Rev}(\mathbf{r}, \mathbf{v}_t) - \max_{\mathbf{r}\in\mathcal{I}_m} \sum_{t=1}^{T} \text{Rev}(\mathbf{r}, \mathbf{v}_t) \leq \frac{Ts}{m}. \tag{4}$$

Setting $m = \sqrt{T}$ and using Theorem 3.4, we obtain the following result for the class of auctions $\mathcal{I}$.

Theorem 3.5. *Consider the online auction design problem for the class of VCG auctions with bidder-specific reserves, $\mathcal{I}$, in s-unit auctions. Let D be the uniform distribution as described in Theorem 2.5. Then, the Oracle-Based Generalized FTPL algorithm with D and datasets that implement $\mathbf{\Gamma}^{\text{VCG}}$ is oracle-efficient with per-round complexity $O(T + n\sqrt{T}\log T)$ and has regret*

$$\mathbb{E}\left[\max_{\mathbf{r}\in\mathcal{I}} \sum_{t=1}^{T} \text{Rev}(\mathbf{r}, \mathbf{v}_t) - \sum_{t=1}^{T} \text{Rev}(\mathbf{r}_t, \mathbf{v}_t)\right] \leq O(ns\sqrt{T}\log T).$$

3.2 Envy-free Item Pricing

In this section, we consider revenue maximization in item pricing [26] in an environment with k heterogeneous items with a supply of $s_\ell \geq 0$ units for each item $\ell \leq k$.

Definition 3.6 (Generalization of Envy-free Item-Pricing Auction). *An* envy-free item-pricing auction *for k heterogeneous items, given supply s_ℓ for $\ell = 1, \ldots, k$, is defined by a vector of prices $\mathbf{a}$, where a_ℓ is the price of item ℓ. The mechanism considers bidders $i = 1, \ldots, n$ in order and allocates to bidder i the bundle $\mathbf{q}_i \in \{0,1\}^k$ that maximizes $v_i(\mathbf{q}_i) - \mathbf{a}\cdot\mathbf{q}_i$, among all feasible bundles, i.e., bundles that can be composed from the remaining supplies. Bidder i is then charged the price $\mathbf{a}\cdot\mathbf{q}_i$.*

Examples of such environments include *unit-demand* bidders and *single-minded* bidders in settings such as *hypergraph pricing*, where bidders seek hyperedges in a hypergraph, and its variant *the highway problem*, where bidders seek hyperedges between sets of contiguous vertices [3, 26].

We emphasize that this is a *generalization* of envy-free item-pricing mechanisms. When the supply is unlimited, any pricing vector is envy-free and the question is centered around finding a revenue-maximizing set of prices. Indeed, several of our results in this section are concerned with the *infinite* supply case where the above item-pricing auctions are known to be envy-free. We also provide no-regret guarantees for revenue maximization in item-pricing auctions with limited supply that satisfy Definition 3.6. However, we note that for some limited-supply settings no item-pricing auction may be envy-free [26].

We represent by $\mathcal{P}_m$ the class of all such item-pricing auctions that satisfy Definition 3.6, where all the prices are strictly positive multiples of $1/m$, i.e., $a_\ell \in \{1/m, \ldots, m/m\}$ for all ℓ. Next, we discuss the construction of an implementable and admissible translation matrix $\mathbf{\Gamma}$. Consider a bid profile where one bidder has value v for bundle $\mathbf{e}_\ell$ and all other bidders have value 0 for all bundles. The revenue of auction $\mathbf{a}$ on such a bid profile is $a_\ell \mathbf{1}_{(v\geq a_\ell)}$. Note the similarity to the case of VCG auctions with bidder-specific reserve prices $\mathbf{r}$, where bid profiles with a single non-zero valuation v_i yielding the revenue $r_i\mathbf{1}_{(v_i\geq r_i)}$ were used to create an implementable construction for $\mathbf{\Gamma}$. We show that a similar construction works for $\mathcal{P}_m$.

Construction of $\mathbf{\Gamma}$: Let $\mathbf{\Gamma}^{\text{IP}}$ be a $|\mathcal{P}_m| \times (k\lceil\log m\rceil)$ binary matrix, where the ℓ^{th} collection of $\lceil\log m\rceil$ columns correspond to the binary encoding of the auction's price for item ℓ. More formally, for any $\ell \leq k$ and $\beta \leq \lceil\log m\rceil$, $\Gamma^{\text{IP}}_{\mathbf{a},j}$ is the β^{th} bit of (the integer) ma_ℓ, where $j = (\ell-1)\lceil\log m\rceil + \beta$.

Next, we show that $\mathbf{\Gamma}^{\text{IP}}$ is admissible and implementable. The proof of the following lemma is analogous to that of Lemma 3.3 and appears in Appendix C.2 for completeness.

Lemma 3.7. *$\mathbf{\Gamma}^{\text{IP}}$ is 1-admissible and implementable with complexity m.*

Our main theorem follows immediately from Lemma 3.7, Theorems 2.5 and 2.10, and the fact that the revenue of the mechanism at every step is at most R. In general, R is at most n.

Theorem 3.8. *Consider the online auction design problem for the class of item-pricing auctions satisfying Definition 3.6, $\mathcal{P}_m$. Let $R = \max_{\mathbf{a},\mathbf{v}} \text{Rev}(\mathbf{a},\mathbf{v})$ and let D be the uniform distribution as described in Theorem 2.5. Then, the Oracle-Based Generalized FTPL algorithm with D and datasets that implement $\mathbf{\Gamma}^{\text{IP}}$ is oracle-efficient with per-round complexity $O(T + km\log m)$ and has regret*

$$\mathbb{E}\left[\max_{\mathbf{a}\in\mathcal{P}_m}\sum_{t=1}^{T}\text{Rev}(\mathbf{a},\mathbf{v}_t) - \sum_{t=1}^{T}\text{Rev}(\mathbf{a}_t,\mathbf{v}_t)\right] \le O\left(kR\sqrt{T}\log m\right).$$

Now consider the class of all envy-free item-pricing auctions where $a_\ell \in [0,1]$ is a real number and denote this class by $\mathcal{P}$. We show that $\mathcal{P}_m$ is a discrete "cover" for $\mathcal{P}$ when there is an *unlimited supply* of all items ($s_\ell = \infty$ for all ℓ) and the bidders have *single-minded* or *unit-demand* valuations. In the single-minded setting, each bidder i is interested in one particular bundle of items $\hat{\mathbf{q}}_i$. That is, $v_i(\mathbf{q}_i) = v_i(\hat{\mathbf{q}}_i)$ for all $\mathbf{q}_i \supseteq \hat{\mathbf{q}}_i$ and 0 otherwise. In the unit-demand setting, each bidder i has valuation $v_i(\mathbf{e}_\ell)$ for item ℓ, and wishes to purchase *at most one item*, i.e., item $\arg\max_\ell (v_i(\mathbf{e}_\ell) - a_\ell)$. We show that in both settings, discretizing item prices cannot decrease the revenue by much (see Appendix C.3).

Lemma 3.9. *For any $\mathbf{a} \in \mathcal{P}$ there is $\mathbf{a}' \in \mathcal{P}_m$, such that for any unit-demand valuation profile $\mathbf{v}$ with infinite supply (the digital goods setting), $\text{Rev}(\mathbf{a},\mathbf{v}) - \text{Rev}(\mathbf{a}',\mathbf{v}) \le nk/m$. Similarly, there is $\mathbf{a}' \in \mathcal{P}_m$, such that for any single-minded valuation profile $\mathbf{v}$ with infinite supply, $\text{Rev}(\mathbf{a},\mathbf{v}) - \text{Rev}(\mathbf{a}',\mathbf{v}) \le nk^2/m$.*

These discretization arguments together with Theorem 3.8 yield the following result for the class of auctions $\mathcal{P}$ (using the fact that $R \le n$, and setting $m = \sqrt{T}$ for the unit-demand and $m = k\sqrt{T}$ for the single-minded setting):

Theorem 3.10. *Consider the online auction design problem for the class of envy-free item-pricing auctions, $\mathcal{P}$, with unit-demand bidders with infinite supply (the digital goods setting). Let D be the uniform distribution as described in Theorem 2.5. Then, the Oracle-Based Generalized FTPL algorithm with D and datasets that implement $\mathbf{\Gamma}^{\text{IP}}$ is oracle-efficient with per-round complexity $O\left(T + k\sqrt{T}\log T\right)$ and has regret*

$$\mathbb{E}\left[\max_{\mathbf{a}\in\mathcal{P}}\sum_{t=1}^{T}\text{Rev}(\mathbf{a},\mathbf{v}_t) - \sum_{t=1}^{T}\text{Rev}(\mathbf{a}_t,\mathbf{v}_t)\right] \le O\left(nk\sqrt{T}\log T\right).$$

Similarly, for single-minded bidders with infinite supply, the Oracle-Based Generalized FTPL algorithm with D and datasets that implement $\mathbf{\Gamma}^{\text{IP}}$ is oracle-efficient with per-round complexity $O\left(T + k^2\sqrt{T}\log(kT)\right)$ and has regret

$$\mathbb{E}\left[\max_{\mathbf{a}\in\mathcal{P}}\sum_{t=1}^{T}\text{Rev}(\mathbf{a},\mathbf{v}_t) - \sum_{t=1}^{T}\text{Rev}(\mathbf{a}_t,\mathbf{v}_t)\right] \le O\left(nk\sqrt{T}\log(kT)\right).$$

3.3 Level Auctions

We next consider the class of *level auctions* introduced by Morgenstern and Roughgarden [43], who show that these auctions can achieve $(1-\epsilon)$-approximate revenue maximization if the valuations

of the bidders are drawn independently (but not necessarily identically) from a distribution, thus approximating Myerson's optimal auction [44]. Using our tools, we derive oracle-efficient no-regret algorithms for this auction class.

The s-level auctions realize a single-item single-unit allocation as follows:

Definition 3.11. *Given $s \geq 2$, an s-level auction $\boldsymbol{\theta}$ is defined by s thresholds for each bidder i, $0 \leq \theta_0^i \leq \cdots \leq \theta_{s-1}^i \leq 1$. For any bid profile $\mathbf{v}$, we let $b_i^{\boldsymbol{\theta}}(v_i)$ denote the largest index b such that $\theta_b^i \leq v_i$, or -1 if $v_i < \theta_0^i$. If $v_i < \theta_0^i$ for all i, the item is not allocated. Otherwise, the item goes to the bidder with the largest index $b_i^{\boldsymbol{\theta}}(v_i)$, breaking ties in favor of bidders with smaller i. The winner pays the price equal to the minimum bid that he could have submitted and still won the item.*

When it is clear from the context, we omit $\boldsymbol{\theta}$ in $b_i^{\boldsymbol{\theta}}(v_i)$ and write just $b_i(v_i)$. In the remainder of the section we assume that $n \geq 2$. For $n = 1$, the level auctions are equivalent to the second-price auctions with reserves, so we can just appeal to the bounds from previous sections (specifically, Theorems 3.4 and 3.5 from Section 3.1).

We consider two classes of s-level auctions, $\mathcal{R}_{s,m}$ and $\mathcal{S}_{s,m}$, where $\mathcal{R}_{s,m}$ is the set of all auctions described by Definition 3.11 with thresholds in the set $\{0, 1/m, \ldots, m/m\}$ and $\mathcal{S}_{s,m}$ is the subset of $\mathcal{R}_{s,m}$ containing the auctions in which the thresholds for each bidder i are distinct.

We first consider $\mathcal{S}_{s,m}$. To construct an admissible and implementable $\boldsymbol{\Gamma}$ for $\mathcal{S}_{s,m}$, we begin with a matrix that is clearly implementable, with each column implemented by a single bid profile, and then show its admissibility.

We consider the bid profiles in which the only non-zero bids are $v_n = \ell/m$ for some $0 \leq \ell \leq m$, and $v_i = 1$ for a single bidder $i < n$. Note that bidder i wins the item in any such profile and pays θ_b^i corresponding to $b = \max\{0, b_n(v_n)\}$. We define a matrix $\boldsymbol{\Gamma}$ with one column for every bid profile of this form and an additional column for the bid profile $\mathbf{e}_n$, with the entries in each row consisting of the revenue of the corresponding auction on the given bid profile. Clearly, $\boldsymbol{\Gamma}$ is implementable. As for admissibility, take $\boldsymbol{\theta} \in \mathcal{S}_{s,m}$ and the corresponding row $\boldsymbol{\Gamma}_{\boldsymbol{\theta}}$. Note that as $v_n = \ell/m$ increases for $\ell = 0, \ldots, m$, there is an increase in $b_n(\ell/m) = -1, 0, \ldots, s-1$, possibly skipping the initial -1. As the level $b_n(v_n)$ increases, the auction revenue attains the values $\theta_0^i, \theta_1^i, \ldots, \theta_{s-1}^i$, changing exactly at those points where v_n crosses thresholds $\theta_1^n, \ldots, \theta_{s-1}^n$. Since any two consecutive thresholds of $\boldsymbol{\theta}$ are different, the thresholds of θ_b^i for $b \geq 0$ and θ_b^n for $b \geq 1$ can be reconstructed by analyzing the revenue of the auction and the values of v_n at which the revenue changes. The remaining threshold θ_0^n is equal to the revenue of the bid profile $\mathbf{v} = \mathbf{e}_n$. Since all of the parameters of the auction can be recovered from the entries in the row $\boldsymbol{\Gamma}_{\boldsymbol{\theta}}$, this shows that any two rows of $\boldsymbol{\Gamma}$ are different and $\boldsymbol{\Gamma}$ is $1/m$-admissible. This reasoning is summarized in the following construction and the corresponding lemma, formally proved in Appendix C.4. See Figure 2 for more intuition.

Construction of $\boldsymbol{\Gamma}$: For $i \in \{1, \ldots, n-1\}$ and $\ell \in \{0, \ldots, m\}$, let $\mathbf{v}^{i,\ell} = \mathbf{e}_i + (\ell/m)\mathbf{e}_n$. Let $V = \{\mathbf{v}^{i,\ell}\}_{i,\ell} \cup \{\mathbf{e}_n\}$. Let $\boldsymbol{\Gamma}^{\mathrm{SL}}$ be the matrix of size $|\mathcal{S}_{s,m}| \times |V|$ with entries indexed by $(\boldsymbol{\theta}, \mathbf{v}) \in \mathcal{S}_{s,m} \times V$, such that $\Gamma^{\mathrm{SL}}_{\boldsymbol{\theta},\mathbf{v}} = \mathrm{Rev}(\boldsymbol{\theta}, \mathbf{v})$.

Lemma 3.12. *$\boldsymbol{\Gamma}^{\mathrm{SL}}$ is $1/m$-admissible and implementable with complexity 1.*

Our next theorem is an immediate consequence of Lemma 3.12, Theorems 2.5 and 2.10, and the fact that the revenue of the mechanism in each round is at most 1.

Theorem 3.13. *Consider the online auction design problem for the class $\mathcal{S}_{s,m}$ of s-level auctions. Let D be the uniform distribution as described in Theorem 2.5. Then, the Oracle-Based Generalized FTPL algorithm with D and datasets that implement $\boldsymbol{\Gamma}^{SL}$ is oracle-efficient with per-round complexity*

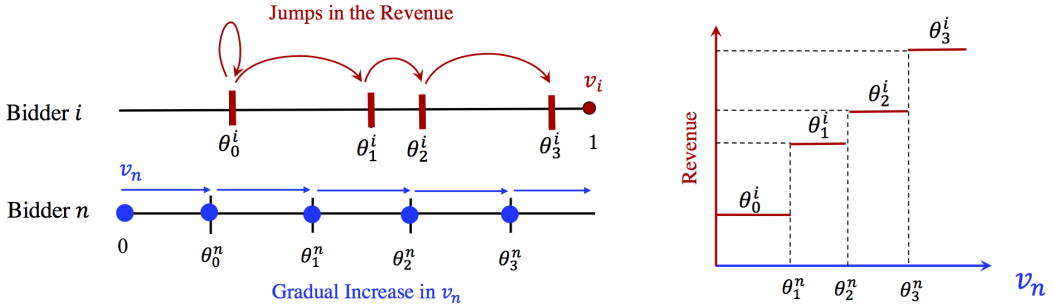

Fig. 2. Demonstration of how $\boldsymbol{\theta}$ can be reconstructed by its revenue on the bid profiles in $V = \{\mathbf{v}^{i,\ell}\}_{i,\ell} \cup \{\mathbf{e}_n\}$. On the left, we show that as the value v_n (blue circle) gradually increases from 0 to 1, the revenue of the auction (red vertical lines) jumps along the sequence of values $\theta_0^i, \theta_1^i, \ldots, \theta_{s-1}^i$. So by analyzing the revenue of an auction on all bid profiles $\{\mathbf{v}^{i,\ell}\}_{i,\ell}$ one can reconstruct $\boldsymbol{\theta}^i$ for $i \neq n$ and $\theta_1^n, \ldots, \theta_{s-1}^n$. To reconstruct θ_0^n, one only needs to consider the profile $\mathbf{e}_n$. The figure on the right demonstrates the revenue of the same auction, where the horizontal axis is the value of v_n and the vertical axis is the revenue of the auction when $v_i = 1$ and all other valuations are 0.

$O(T + nm)$ and has regret

$$\mathbb{E}\left[\max_{\boldsymbol{\theta}\in\mathcal{S}_{s,m}} \sum_{t=1}^{T} \mathrm{Rev}(\boldsymbol{\theta}, \mathbf{v}_t) - \sum_{t=1}^{T} \mathrm{Rev}(\boldsymbol{\theta}_t, \mathbf{v}_t)\right] \leq O(nm^2\sqrt{T}).$$

Next, we turn our attention to the class of auctions $\mathcal{R}_{s,m}$ and construct an admissible and implementable matrix for this auction class. As a warm-up, we demonstrate how the structure of $\mathcal{R}_{s,m}$ differs from $\mathcal{S}_{s,m}$ and argue that $\boldsymbol{\Gamma}^{\mathrm{SL}}$ is not admissible for $\mathcal{R}_{s,m}$. Take $\mathbf{v}$ such that $v_i = 1$ and v_n is some multiple of $1/m$. Similarly as before, as v_n increases there is also an increase in $b_n(v_n)$ and the revenue $\theta^i_{b_n(v_n)}$. However, the thresholds of bidder n are no longer required to be distinct, so $b_n(v_n)$ can skip certain values, and as a result θ_b^i is not revealed for all b. In addition, when the thresholds of bidder i are not distinct, we have that even when $b_n(v_n)$ strictly increases, there might be no increase in the revenue, and as a result θ_b^n cannot be reconstructed.

We next show how to construct an admissible matrix for $\mathcal{R}_{s,m}$. In preparation for this construction, we first define an equivalence relation among auctions which puts auctions with distinct parameters but identical outcomes in the same equivalence class. Specifically, we say that two auctions $\boldsymbol{\theta}$ and $\boldsymbol{\theta}'$ are *equivalent* if for any bid profile $\mathbf{v}$, $\mathrm{Rev}(\boldsymbol{\theta}, \mathbf{v}) = \mathrm{Rev}(\boldsymbol{\theta}', \mathbf{v})$, i.e., they receive the same revenue on all bid profiles. Note that it is sufficient to limit our attention to a subset of auctions $\bar{\mathcal{R}}_{s,m} \subseteq \mathcal{R}_{s,m}$ that includes exactly one auction from each equivalence class. The regret with respect to $\bar{\mathcal{R}}_{s,m}$ is the same as the regret with respect to $\mathcal{R}_{s,m}$. Also, when applying Algorithm 2, any result x_t returned by an offline oracle for $\mathcal{R}_{s,m}$ can be replaced by the equivalent $\bar{x}_t \in \bar{\mathcal{R}}_{s,m}$ without affecting the algorithm's regret. This means that an offline oracle for $\mathcal{R}_{s,m}$ can be viewed as an offline oracle for $\bar{\mathcal{R}}_{s,m}$ and used together with an admissible and implementable matrix $\boldsymbol{\Gamma}^{\overline{\mathrm{RL}}}$ for the auction class $\bar{\mathcal{R}}_{s,m}$ to solve the online optimization problem for the larger but equivalent auction class $\mathcal{R}_{s,m}$. In the following, we introduce an implementable and admissible construction for $\bar{\mathcal{R}}_{s,m}$.

Construction of $\boldsymbol{\Gamma}$: Let $V = \{\mathbf{v} \mid \mathbf{v} \in \{0, 1/m, \ldots, m/m\}^n \text{ and } \|\mathbf{v}\|_0 \leq 3\}$, where $\|\cdot\|_0$ denotes the number of non-zero entries of a vector. Let $\boldsymbol{\Gamma}^{\overline{\mathrm{RL}}}$ be the matrix of size $|\bar{\mathcal{R}}_{s,m}| \times |V|$ with entries indexed by $(\boldsymbol{\theta}, \mathbf{v}) \in \bar{\mathcal{R}}_{s,m} \times V$, such that $\Gamma^{\overline{\mathrm{RL}}}_{\boldsymbol{\theta},\mathbf{v}} = \mathrm{Rev}(\boldsymbol{\theta}, \mathbf{v})$.

Lemma 3.14. *$\boldsymbol{\Gamma}^{\overline{\mathrm{RL}}}$ is $1/m$-admissible and implementable with complexity 1 for the class of auctions $\bar{\mathcal{R}}_{s,m}$.*

Proof. Clearly, $\boldsymbol{\Gamma}^{\overline{\mathrm{RL}}}$ is implementable with complexity 1 for the class of auctions $\bar{\mathcal{R}}_{s,m}$. We show that $\boldsymbol{\Gamma}^{\overline{\mathrm{RL}}}$ is also admissible. Take any two auctions $\boldsymbol{\theta}$ and $\boldsymbol{\theta}'$ in $\bar{\mathcal{R}}_{s,m}$. Since these auctions are not equivalent, there is a bid profile $\mathbf{v}$ such that $\mathrm{Rev}(\boldsymbol{\theta},\mathbf{v}) \neq \mathrm{Rev}(\boldsymbol{\theta}',\mathbf{v})$. Without loss of generality, assume that $\mathrm{Rev}(\boldsymbol{\theta},\mathbf{v}) < \mathrm{Rev}(\boldsymbol{\theta}',\mathbf{v})$. In what follows, we construct a corresponding bid profile $\mathbf{v}' \in V$ such that $\mathrm{Rev}(\boldsymbol{\theta},\mathbf{v}') \neq \mathrm{Rev}(\boldsymbol{\theta}',\mathbf{v}')$. Let i and i' be the winners in the auctions $\boldsymbol{\theta}$ and $\boldsymbol{\theta}'$, respectively. Let j and j' be the bidders with second highest bucketed bids, i.e., the bidders who set prices in auctions $\boldsymbol{\theta}$ and $\boldsymbol{\theta}'$, respectively. We consider two cases:

If i, i', j, and j' are not distinct, we construct the bid profile $\mathbf{v}'$ that is the same as $\mathbf{v}$ on indices i, i', j, and j', and is 0 otherwise. Note that $\mathbf{v}'$ has at most 3 non-zero elements, hence $\mathbf{v}' \in V$. Moreover, the bids of the winners and the price setters are the same in both bid profiles, so the allocations and payments of both auctions remain the same. That is,

$$\mathrm{Rev}(\boldsymbol{\theta},\mathbf{v}') = \mathrm{Rev}(\boldsymbol{\theta},\mathbf{v}) \neq \mathrm{Rev}(\boldsymbol{\theta}',\mathbf{v}) = \mathrm{Rev}(\boldsymbol{\theta}',\mathbf{v}').$$

If i, i', j, and j' are all distinct, we construct the bid profile $\mathbf{v}' \in V$ that is the same as $\mathbf{v}$ on indices i, i', and j', and is 0 otherwise (including on index j). On auction $\boldsymbol{\theta}'$, since the bids of the winner and the price setter, i' and j' are the same in both bid profiles, the allocation and payment of both auctions remain the same, and we have $\mathrm{Rev}(\boldsymbol{\theta}',\mathbf{v}') = \mathrm{Rev}(\boldsymbol{\theta}',\mathbf{v})$. On auction $\boldsymbol{\theta}$, the winner's bid remains the same and the price setter's bid is equal or lowered, so $\mathrm{Rev}(\boldsymbol{\theta},\mathbf{v}') \leq \mathrm{Rev}(\boldsymbol{\theta},\mathbf{v})$. Hence,

$$\mathrm{Rev}(\boldsymbol{\theta},\mathbf{v}') \leq \mathrm{Rev}(\boldsymbol{\theta},\mathbf{v}) < \mathrm{Rev}(\boldsymbol{\theta}',\mathbf{v}) = \mathrm{Rev}(\boldsymbol{\theta}',\mathbf{v}').$$

Since the revenue of any auction is a multiple of $1/m$ and any two rows of $\boldsymbol{\Gamma}^{\overline{\mathrm{RL}}}$ differ in at least one entry by $1/m$, we obtain that $\boldsymbol{\Gamma}^{\overline{\mathrm{RL}}}$ is $1/m$-admissible. □

Our next theorem is an immediate consequence of Lemma 3.14, Theorems 2.5 and 2.10, the equality of regrets with respect to $\bar{\mathcal{R}}_{s,m}$ and $\mathcal{R}_{s,m}$, and the fact that the revenue of the mechanism in each round is at most 1. As discussed above, Algorithm 2 can use an offline oracle for $\mathcal{R}_{s,m}$ in place of an offline oracle for $\bar{\mathcal{R}}_{s,m}$.

Theorem 3.15. *Consider the online auction design problem for the class $\mathcal{R}_{s,m}$ of s-level auctions. Let D be the uniform distribution as described in Theorem 2.5. Then, the Oracle-Based Generalized FTPL algorithm with D and datasets that implement $\boldsymbol{\Gamma}^{\overline{\mathrm{RL}}}$ is oracle-efficient with per-round complexity $O(T + n^3m^3)$ and has regret*

$$\mathbb{E}\left[\max_{\boldsymbol{\theta}\in\mathcal{R}_{s,m}} \sum_{t=1}^{T} \mathrm{Rev}(\boldsymbol{\theta},\mathbf{v}_t) - \sum_{t=1}^{T} \mathrm{Rev}(\boldsymbol{\theta}_t,\mathbf{v}_t)\right] \leq O(n^3m^4\sqrt{T}).$$

4 STOCHASTIC ADVERSARIES AND UNIVERSAL BENCHMARKS

So far our results apply to general adversaries, where the sequence of adversary actions is arbitrary, and where we can achieve the payoff that is close to the payoff of the best action in our class. For auctions, we might be interested in the comparison with an arbitrary auction rather than an auction that is in our class. In this section, we show that in certain cases this can be achieved if we impose distributional assumptions on the sequence of the adversary.

We start with the easier setting where the actions of the adversary are drawn i.i.d. across all rounds and then we analyze the slightly more complex setting where the actions of the adversary follow a fast-mixing Markov chain. For both settings we show that the average payoff of the learning algorithm is close to the optimal expected payoff, where expectation is with respect to

the adversary's distribution in the i.i.d. setting and the adversary's stationary distribution in the Markovian setting.

In online auction design, we can combine these results with approximate optimality results of simple auctions, such as s-level auctions or VCG with bidder-specific reserves, to prove universal optimality of our online learning algorithms in these distributional settings rather than only the optimality among the auctions within the class over which our algorithms are learning.

4.1 Stochastic Adversaries

I.I.D. Adversary. When the adversary actions are drawn independently from the same unknown distribution F, we can use Chernoff-Hoeffding bound to show that the average payoff of a no-regret learner converges to the best payoff one could achieve in expectation over the distribution F (see Appendix D.1 for the proof):

Lemma 4.1. *Suppose that $y_1, \ldots, y_T$ are i.i.d. draws from a distribution F. Then for any no-regret learning algorithm, with probability at least $1 - \xi$,*

$$\frac{1}{T}\sum_{t=1}^{T} \mathbb{E}_{x_t}[f(x_t, y_t)] \geq \sup_{x \in \mathcal{X}} \mathbb{E}_{y\sim F}[f(x, y)] - \sqrt{\frac{\log(1/\xi)}{2T}} - \frac{\textsc{Regret}}{T}.$$

Markovian Adversary. A natural intermediate model between i.i.d valuations and fully adversarial valuations is the *Markovian* adversary, also called the *Markovian environment*. In Markovian environments, the adversary's actions (corresponding to valuations) can be correlated across time, but they must satisfy the Markov assumption: the adversary action y_t conditioned on y_{t-1} must be conditionally independent of all the preceding actions $y_{t'}$, $t' \leq t-2$. This captures, for example, auction environments where with some probability ρ the valuation in a given round is the same as in the previous round, and with probability $(1-\rho)$ it is a fresh draw from some fixed distribution.

Suppose that the choice of the adversary y_t follows a stationary and reversible Markov process based on some transition matrix $P(y, y')$ with a stationary distribution F. Moreover, consider the case where the set $\mathcal{Y}$ is finite. For any Markov chain, the spectral gap γ is defined as the difference between the first and the second largest eigenvalue of the transition matrix P (the first eigenvalue always being 1). We will assume that this gap is bounded away from zero. The spectral gap of a Markov chain is strongly related to its mixing time. In this work we will specifically use the following result of Paulin [46], which is a Bernstein concentration inequality for sums of dependent random variables sampled from a stationary Markov chain with spectral gap bounded away from zero. A Markov chain $y_1, \ldots, y_T$ is stationary if $y_1 \sim F$ where F is the stationary distribution, and is reversible if for any y, y', $F(y)P(y, y') = F(y')P(y', y)$. For simplicity, we focus on stationary chains, though similar results hold for non-stationary chains (see Paulin [46] and references therein).

Theorem 4.2 (Paulin [46], Theorem 3.8). *Let $X_1, \ldots, X_T$ be a stationary and reversible Markov chain on a state space Ω, with stationary distribution F and spectral gap γ. Let $g : \Omega \to [0, 1]$, then*

$$\Pr\left[\left|\frac{1}{T}\sum_{t=1}^{T} g(X_t) - \mathbb{E}_{X\sim F}[g(X)]\right| > \epsilon\right] \leq 2\exp\left(-\frac{T\gamma\epsilon^2}{4 + 10\epsilon}\right).$$

Applying this result, we obtain the following lemma (see Appendix D.2 for the proof):

Lemma 4.3. *Suppose that the adversary actions $y_1, \ldots, y_T$ form a stationary and reversible Markov chain with stationary distribution F and spectral gap γ. Then for any no-regret learning algorithm,*

with probability at least $1-\xi$,

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}_{x_t}[f(x_t,y_t)] \geq \sup_{x\in\mathcal{X}}\mathbb{E}_{y\sim F}[f(x,y)] - \sqrt{\frac{14\log(2/\xi)}{\gamma T}} - \frac{\textsc{Regret}}{T}.$$

Example 4.4 (Sticky Markov Chain). *Consider a Markov chain where at each round* y_t *is equal to* y_{t-1} *with some probability* $\rho \geq 1/2$ *and with the remaining probability* $(1-\rho)$ *it is drawn independently from some fixed distribution* F*. It is clear that the stationary distribution of this chain is equal to* F*. We can bound the spectral gap of this Markov chain by the Cheeger bound [11]. The Cheeger constant for a finite state, reversible Markov chain is defined, and in this case bounded, as*

$$\begin{aligned}\Phi &= \min_{Q\subseteq\Omega:F(Q)\leq 1/2}\frac{\sum_{y\in Q}\sum_{y'\in Q^c}F(y)P(y,y')}{F(Q)} = \min_{Q\subseteq\Omega:F(Q)\leq 1/2}\frac{\sum_{y\in Q}\sum_{y'\in Q^c}F(y)(1-\rho)F(y')}{F(Q)}\\ &= \min_{Q\subseteq\Omega:F(Q)\leq 1/2}(1-\rho)\frac{F(Q)\cdot F(Q^c)}{F(Q)} = \min_{Q\subseteq\Omega:F(Q)\leq 1/2}(1-\rho)F(Q^c) \geq \frac{1-\rho}{2}\quad.\end{aligned}$$

Moreover, by the Cheeger bound we know that $\gamma \geq \frac{\Phi^2}{2} \geq \frac{(1-\rho)^2}{8}$*. Thus we get that for such a sequence of adversary actions, with probability* $1-\xi$,

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}_{x_t}[f(x_t,y_t)] \geq \sup_{x\in\mathcal{X}}\mathbb{E}_{y\sim F}[f(x,y)] - \frac{4}{1-\rho}\sqrt{\frac{7\log(2/\xi)}{T}} - \frac{\textsc{Regret}}{T}\quad.$$

4.2 Implications for Online Optimal Auction Design

Consider the online optimal auction design problem for a single item and n bidders. Suppose that the adversary who picks the valuation profiles $\mathbf{v}_1,\ldots,\mathbf{v}_T$ is Markovian and that the stationary distribution F of the chain is independent across players, i.e., the stationary distribution is a product distribution $F = F_1\times\cdots\times F_n$. Then we know that the optimal auction for this setting is *Myerson's (optimal) auction* [44], which translates the players' values based on some monotone function ϕ, known as the *ironed virtual value function*, and then allocates the item to the bidder with the highest virtual value, charging payments so that the mechanism is dominant-strategy truthful. In this section, we use the Generalized FTPL algorithm to compete with Myerson's optimal auction in this Markovian environment. This extends the prior work that only gave such guarantees for the i.i.d. setting.[7]

A natural approach for approximating the overall optimal auction in a Markovian environment is through level auctions. As discussed in Section 3.3, Morgenstern and Roughgarden [43] show that level auctions with repeated thresholds and arbitrarily fine discretization approximate Myerson's optimal auction in terms of revenue. In more detail, when distributions F_i are bounded in $[1,H]$, the class of auctions $\mathcal{R}_{s,\infty}$ with $s = \Omega\left(\frac{1}{\epsilon}+\log_{1+\epsilon}H\right)$ achieves expected revenue of at least a factor of $(1-\epsilon)$ of the expected optimal revenue of Myerson's auction. Since the runtime and regret of our Generalized FTPL algorithm for the class of auctions $\mathcal{R}_{s,m}$ scale with the discretization level m, we require m to be finite. Therefore, we cannot use this characterization of Morgenstern and Roughgarden [43] directly. Analogously to these results, we prove and use an additive approximation guarantee for the class of discretized level auctions $\mathcal{R}_{n/\epsilon,1/\epsilon}$, showing that

$$\max_{\boldsymbol{\theta}\in\mathcal{R}_{n/\epsilon,1/\epsilon}}\mathbb{E}_{\mathbf{v}\sim F}\left[\mathrm{Rev}(\boldsymbol{\theta},\mathbf{v})\right] \geq \textsc{Opt}(F) - \epsilon, \tag{5}$$

[7] If we know ahead of the time that the stationary distribution is symmetric across bidders, i.e., $F_1 = F_2 = \cdots = F_n$, then the optimal auction is a second-price auction with a reserve. In that case, we can appeal to Theorem 3.5 to obtain a better regret bound than those presented in this section.

where $\textsc{Opt}(F)$ is the optimal revenue achievable by any dominant-strategy truthful mechanism for valuation vector distribution F.

At a high level, we prove Equation (5) using a three-step approach. We first consider the discretization of the bidders' valuations. That is, for any bid profile $\mathbf{v}$, we consider $\lfloor \mathbf{v} \rfloor_\epsilon$ that denotes the bid profile where each entry of $\mathbf{v}$ is rounded down to the nearest whole multiple of ϵ. As the first step, it is not hard to show that for any $\boldsymbol{\theta} \in \mathcal{R}_{n/\epsilon,1/\epsilon}$, $\text{Rev}(\boldsymbol{\theta}, \lfloor \mathbf{v} \rfloor_\epsilon) = \text{Rev}(\boldsymbol{\theta}, \mathbf{v})$. In the second step, we show that Myerson's optimal auction on the discretized valuations indeed belongs to the class $\mathcal{R}_{n/\epsilon,1/\epsilon}$. That is, when F' represents the distribution over valuations $\lfloor \mathbf{v} \rfloor_\epsilon$, we have

$$\max_{\boldsymbol{\theta} \in \mathcal{R}_{n/\epsilon,1/\epsilon}} \mathbb{E}_{\mathbf{v} \sim F} \left[\text{Rev}(\boldsymbol{\theta}, \lfloor \mathbf{v} \rfloor_\epsilon) \right] = \textsc{Opt}(F').$$

To prove this we use a characterization of Myerson's optimal auction on discrete (and not necessarily regular) distributions provided by Elkind [22]. Much like level auctions, this characterization assigns a bucket number to each possible valuation of each bidder and allocates the item to the valuation with the highest bucket number. We show that one can create n/ϵ thresholds for each bidder to mimic this bucketing effect. Therefore, the class of auctions $\mathcal{R}_{n/\epsilon,1/\epsilon}$ includes an optimal auction for the discretized valuations. In the last step, we use a result of Devanur et al. [17] that establishes that $\textsc{Opt}(F') \geq \textsc{Opt}(F) - \epsilon$. Putting the above ingredients together, Theorem 4.5 shows that the Genearlized FTPL algorithm for $\mathcal{R}_{n/\epsilon,1/\epsilon}$ approximates Myerson's auction for a given Markov chain. We defer the detailed proof of Theorem 4.5 to Section 4.3 after giving an example of this setting.

Theorem 4.5 (Competing with Universally Optimal Auction). *Consider the online auction design problem for a single item among n bidders, where the sequence of valuation vectors $\mathbf{v}_1, \ldots, \mathbf{v}_T$ is Markovian, following a stationary and reversible Markov process with a spectral gap of $\gamma > 0$ and with a stationary distribution F which is a product distribution across bidders, meaning $F = F_1 \times \cdots \times F_n$. Then, the Generalized FTPL algorithm for $\mathcal{R}_{n/\epsilon,1/\epsilon}$, where $\epsilon = \Theta(n^{3/5} T^{-1/10})$, guarantees the following bound with probability at least $1 - \xi$:*

$$\frac{1}{T} \sum_{t=1}^{T} \mathbb{E} \left[\text{Rev}(\boldsymbol{\theta}_t, \mathbf{v}_t) \right] \geq \textsc{Opt}(F) - \sqrt{\frac{14 \log(2/\xi)}{\gamma T}} - O\left(n^{3/5} T^{-1/10} \right).$$

Example 4.6 (Valuation Shocks). *Consider the setting where valuations of players in the beginning of time are drawn from some product distribution $F = F_1 \times \cdots \times F_n$. Then in each round with some probability ρ the valuations of all players remain the same as in the previous round, while with some probability $1 - \rho$, there is a shock in the market and the valuations of the players are re-drawn from distribution F. As we analyzed in the previous section, the spectral gap of the Markov chain defined by this problem is at least $\frac{(1-\rho)^2}{8}$. Thus we get a regret bound which depends inversely on the quantity $1 - \rho$.*

Hence, our online learning algorithm achieves revenue that is close to the optimal revenue achievable by any dominant-strategy truthful mechanism for the distribution F. More importantly, it achieves this guarantee even if the valuations of the players are not drawn i.i.d. at every iteration and even if the learner does not know what the distribution F is, or when the valuations of the players are going to be re-drawn, or what the rate ρ of shocks in the markets is.

4.3 Proof of Theorem 4.5

We set out to prove that

$$\frac{1}{T} \sum_{t=1}^{T} \mathbb{E} \left[\text{Rev}(\boldsymbol{\theta}_t, \mathbf{v}_t) \right] \geq \textsc{Opt}(F) - \epsilon - \sqrt{\frac{14 \log(2/\xi)}{\gamma T}} - O\left(n^{3/5} T^{-1/10} \right).$$

We first show that $\mathrm{Rev}(\boldsymbol{\theta}, \mathbf{v}) = \mathrm{Rev}(\boldsymbol{\theta}, \lfloor \mathbf{v} \rfloor_\epsilon)$ for any $\mathbf{v}$ and any $\boldsymbol{\theta} \in \mathcal{R}_{n/\epsilon,1/\epsilon}$. To start, note that the thresholds in $\boldsymbol{\theta}$ are multiples of ϵ, so bid profiles $\mathbf{v}$ and $\lfloor \mathbf{v} \rfloor_\epsilon$ are bucketed identically by $\boldsymbol{\theta}$. Therefore, these two bid profiles have the same winner and the same payment amount (equal to the smallest bid with which the bidder still wins the item). Thus, we have that $\mathrm{Rev}(\boldsymbol{\theta}, \mathbf{v}) = \mathrm{Rev}(\boldsymbol{\theta}, \lfloor \mathbf{v} \rfloor_\epsilon)$, and by Lemma 4.3, we can bound the revenue of the Generalized FTPL algorithm as

$$\frac{1}{T}\sum_{t=1}^{T} \mathbb{E}_{\theta_t}\Big[\mathrm{Rev}(\boldsymbol{\theta}_t, \mathbf{v}_t)\Big] \geq \sup_{\boldsymbol{\theta} \in \mathcal{R}_{n/\epsilon,1/\epsilon}} \mathbb{E}_{\mathbf{v}\sim F}\Big[\mathrm{Rev}(\boldsymbol{\theta}, \mathbf{v})\Big] - \sqrt{\frac{14\log(2/\xi)}{\gamma T}} - O\left(n^{3/5}T^{-1/10}\right)$$

$$\geq \sup_{\boldsymbol{\theta} \in \mathcal{R}_{n/\epsilon,1/\epsilon}} \mathbb{E}_{\mathbf{v}\sim F}\Big[\mathrm{Rev}\Big(\boldsymbol{\theta}, \lfloor \mathbf{v} \rfloor_\epsilon\Big)\Big] - \sqrt{\frac{14\log(2/\xi)}{\gamma T}} - O\left(n^{3/5}T^{-1/10}\right). \quad (6)$$

Next, we show that the expected revenue of the optimal auction $\boldsymbol{\theta} \in \mathcal{R}_{n/\epsilon,1/\epsilon}$ on the rounded valuation $\mathbf{v}' = \lfloor \mathbf{v} \rfloor_\epsilon$ is close to $\textsc{Opt}(F')$, where F' is the product distribution of $\mathbf{v}'$.

Lemma 4.7. *Let $F' = F'_1 \times \cdots \times F'_n$ be a product distribution over $\mathbf{v}'$ that are supported on multiples of ϵ. Let $\textsc{Opt}(F')$ be the revenue of Myerson's optimal auction on F'. Then,*

$$\max_{\boldsymbol{\theta} \in \mathcal{R}_{n/\epsilon,1/\epsilon}} \mathbb{E}_{\mathbf{v}'\sim F'}\left[\mathrm{Rev}(\boldsymbol{\theta}, \mathbf{v}')\right] = \textsc{Opt}(F').$$

To prove Lemma 4.7, we first use the following result of Elkind [22] which describes Myerson's optimal auctions on arbitrary discrete distributions with a finite support.

Lemma 4.8 (Theorem 3.2 of Elkind [22]). *Consider a product distribution $F' = F'_1 \times \cdots \times F'_n$, where each F'_i is a discrete distribution supported on $m+1$ values indexed in an increasing order by $\ell \in \{0, 1, \ldots, m\}$. There exist values $\bar{c}_i^\ell$ associated with all possible bid levels of each bidder, satisfying $\bar{c}_i^\ell \leq \bar{c}_i^{\ell+1}$ for all $i \in [n]$ and $\ell \in \{0, 1, \ldots, m-1\}$, giving rise to the optimal auction for F' according to the following mechanism: Sort bidders in the order of decreasing values $\bar{c}_i^\ell$ that correspond to their submitted bids, breaking ties in favor of bidders with smaller indices. Allocate the item to the first bidder whose corresponding $\bar{c}_i^\ell$ is non-negative and charge the bidder the minimum bid that he could have placed and still won the item.*

Proof of Lemma 4.7. Since each bid is of the form $\ell\epsilon$ where $\ell \in \{0, \ldots, m\}$, by Lemma 4.8, there exist suitable values $\bar{c}_i^\ell$ that implement the optimal auction. We will show that there is a level auction $\boldsymbol{\theta} \in \mathcal{R}_{n/\epsilon,1/\epsilon}$ that implements the same auction.

First sort the values $\bar{c}_i^\ell$ from smallest to largest, breaking ties in favor of smaller ℓ and larger i (same as done in Elkind's mechanism, but in the reverse order). Let $\mathrm{pos}(i, \ell)$ be as follows: Assign $\mathrm{pos}(i, \ell) = -1$ for all leading negative $\bar{c}_i^\ell$ and then let $\mathrm{pos}(i, \ell)$ be the position of $\bar{c}_i^\ell$ in this list, starting with index 0 after the leading -1s. The allocation rule of Elkind's mechanism is equivalent to replacing each bid by its corresponding $\mathrm{pos}(i, \ell)$ and giving the item to the bidder with the highest non-negative position. We now inductively construct a level-auction mechanism $\boldsymbol{\theta}$ in which $b_i^{\boldsymbol{\theta}}(\ell\epsilon) = \mathrm{pos}(i, \ell)$.

Take any $i \in [n]$. We begin with an empty set of thresholds and introduce new thresholds gradually for $\ell = 0, \ldots, m$. For ease of exposition let $\mathrm{pos}(i, -1) = -1$ and $b_i^{\boldsymbol{\theta}}(-\epsilon) = -1$. This guarantees that $b_i^{\boldsymbol{\theta}}(\ell\epsilon) = \mathrm{pos}(i, \ell)$ holds for $\ell = -1$. Now assume that the thresholds introduced so far guarantee that $b_i^{\boldsymbol{\theta}}(\ell\epsilon) = \mathrm{pos}(i, \ell)$ holds for $\ell < k$. For $\ell = k$, we only create additional thresholds with the value k/m, which will preserve this property for $\ell < k$, but also allows us to extend it to $\ell = k$. Specifically, we add $\mathrm{pos}(i, k) - \mathrm{pos}(i, k-1)$ many thresholds with the value k/m. Using the induction hypothesis, we have

$$b_i^{\boldsymbol{\theta}}(k\epsilon) = b_i^{\boldsymbol{\theta}}\Big((k-1)\epsilon\Big) + \mathrm{pos}(i, k) - \mathrm{pos}(i, k-1) = \mathrm{pos}(i, k).$$

Hence, by induction $b_i^{\theta}(\ell\epsilon) = \text{pos}(i,\ell)$ for all ℓ. This completes the claim that we have created a level auction $\theta \in \mathcal{R}_{n/\epsilon,1/\epsilon}$ that implements the optimal auction described in Lemma 4.8. □

We next use the result of Devanur et al. [17] that shows that $\textsc{Opt}(F')$ is almost as large as $\textsc{Opt}(F)$.

Lemma 4.9 (Lemma 4.3 of Devanur et al. [16]). *Given any product distribution $F = F_1 \times \cdots \times F_n$, let F' be the distribution obtained by rounding down the values from F to the nearest multiple of ϵ. Then $\textsc{Opt}(F') \geq \textsc{Opt}(F) - \epsilon$.*

Combining the above proves Theorem 4.5.

5 CONTEXTUAL ONLINE LEARNING: LEARNING WITH SIDE INFORMATION

We now consider a generalization of the online learning setting of Section 2, where in each round the learner also observes some side information, called *context*. The context σ_t in round t comes from some abstract context space Σ. The learner wants to use this contextual information to improve his performance. Specifically, the learner's goal is to compete with a set of policies Π, where each policy $\pi \in \Pi$ is a mapping from contexts $\sigma \in \Sigma$ to actions $\pi(\sigma) \in \mathcal{X}$.

The adversary, in each round, chooses both a context $\sigma_t \in \Sigma$ and an action $y_t \in \mathcal{Y}$. The payoff of the learner if he chooses a policy π_t is then $f(\pi_t(\sigma_t), y_t)$. The regret of an algorithm is given by:

$$\textsc{Regret} = \mathbb{E}\left[\max_{\pi\in\Pi} \sum_{t=1}^{T} f(\pi(\sigma_t), y_t) - \sum_{t=1}^{T} f(\pi_t(\sigma_t), y_t)\right].$$

An offline oracle in the contextual setting is an algorithm that takes as input a distribution over pairs of contexts and adversary actions and returns the best policy in the policy space Π for this distribution.

The contextual learning problem, described by $\mathcal{X}$, Σ, Π, $\mathcal{Y}$ and f, can be viewed as a specific instance of a (non-contextual) learning problem described in Section 2, with the learner's action space $\mathcal{X}_c = \Pi$, adversary action space $\mathcal{Y}_c = \Sigma \times \mathcal{Y}$ and payoff function $f_c(\pi, (\sigma, y)) = f(\pi(\sigma), y)$. Moreover, any offline oracle for the contextual problem $(\mathcal{X}, \Sigma, \Pi, \mathcal{Y}, f)$ is also an offline oracle for the non-contextual problem $(\mathcal{X}_c, \mathcal{Y}_c, f_c)$. Below, we show how to obtain oracle-efficient no-regret algorithms for $(\mathcal{X}_c, \mathcal{Y}_c, f_c)$ if we have access to an admissible and implementable matrix $\mathbf{\Gamma}$ for the simpler non-contextual problem $(\mathcal{X}, \mathcal{Y}, f)$. We will conflate the problems $(\mathcal{X}, \Sigma, \Pi, \mathcal{Y}, f)$ and $(\mathcal{X}_c, \mathcal{Y}_c, f_c)$, and jointly refer to them as the *contextual problem*, and refer to the simpler problem $(\mathcal{X}, \mathcal{Y}, f)$ as the *non-contextual problem*.

Example 5.1 (Contextual Online Auction Design). *In contextual online auction design, the auctioneer gets to see some side information about the bidders before they place their bids. The goal of the auctioneer is to compete with a set of policies that map such contextual information to an auction. Given a class of auctions $\mathcal{A}$, a set of possible valuations $\mathcal{V}$, a policy class Π and an unknown sequence of contexts and bid profiles $(\sigma_1, \mathbf{v}_1), \ldots, (\sigma_T, \mathbf{v}_T) \in \Sigma \times \mathcal{V}^n$, the goal of an online algorithm is to pick in each round a policy $\pi_t \in \Pi$ such that the algorithm's total revenue is close to the revenue of the best policy $\pi \in \Pi$ in hindsight:*

$$\textsc{Regret} = \mathbb{E}\left[\max_{\pi\in\Pi} \sum_{t=1}^{T} \text{Rev}(\pi(\sigma_t), \mathbf{v}_t) - \sum_{t=1}^{T} \text{Rev}(\pi_t(\sigma_t), \mathbf{v}_t)\right] \leq o(T).$$

5.1 From Non-Contextual to Contextual Learning for Separable Policy Spaces

Assume we are given an admissible and implementable matrix $\mathbf{\Gamma}$ for $(\mathcal{X}, \mathcal{Y}, f)$. We next show how to construct an admissible and implementable matrix $\mathbf{\Gamma}^Q$ for the contextual problem $(\mathcal{X}_c, \mathcal{Y}_c, f_c)$. Our construction builds on the notion of a separator of a policy space, also known as the *universal*

identification sequence, introduced by Goldman et al. [25]; the term *separator* is due to Syrgkanis et al. [53].

Definition 5.2 (Separator). *A set $Q \subseteq \Sigma$ of contexts is a separator for a policy space Π, if for any two policies $\pi, \pi' \in \Pi$, there exists a context $\sigma \in Q$ such that $\pi(\sigma) \neq \pi'(\sigma)$.*

Definition 5.3 (Contextual Q-extension of a matrix $\mathbf{\Gamma}$). *For any matrix $\mathbf{\Gamma} \in [0,1]^{|\mathcal{X}| \times N}$, we define its contextual Q-extension $\mathbf{\Gamma}^Q$ as an $|\Pi| \times (|Q| \cdot N)$ matrix, where each column j_c is associated with a pair (σ, j) for $\sigma \in Q$ and $j \in [N]$ and the entry of $\mathbf{\Gamma}^Q$ at coordinates $(\pi, (\sigma, j))$ is equal to the entry of matrix $\mathbf{\Gamma}$ at coordinates $(\pi(\sigma), j)$, i.e.,*

$$\Gamma^Q_{\pi,(\sigma,j)} = \Gamma_{\pi(\sigma),j}. \tag{7}$$

The next two lemmas, whose proofs appear in Appendices E.1 and E.2, show that if a matrix is admissible and implementable for $(\mathcal{X}, \mathcal{Y}, f)$, then its contextual extension with respect to a separator is admissible and implementable for $(\mathcal{X}_c, \mathcal{Y}_c, f_c)$.

Lemma 5.4 (Contextual Admissibility). *If $\mathbf{\Gamma}$ is a δ-admissible matrix for $(\mathcal{X}, \mathcal{Y}, f)$ and Q is a separator for the policy space Π, then the contextual Q-extension $\mathbf{\Gamma}^Q$ is δ-admissible for the contextual problem $(\mathcal{X}_c, \mathcal{Y}_c, f_c)$.*

Lemma 5.5 (Contextual Implementability). *If matrix $\mathbf{\Gamma}$ is implementable with complexity M for $(\mathcal{X}, \mathcal{Y}, f)$, then matrix $\mathbf{\Gamma}^Q$ is implementable with complexity M for the contextual problem $(\mathcal{X}_c, \mathcal{Y}_c, f_c)$.*

Given the above two lemmas, we can now invoke Corollary 2.11 to obtain oracle-efficient online learning for the contextual problem $(\mathcal{X}_c, \mathcal{Y}_c, f_c)$:

Corollary 5.6. *Assume that $\mathbf{\Gamma} \in [0,1]^{|\mathcal{X}| \times N}$ is implementable with complexity M and δ-admissible for $(\mathcal{X}, \mathcal{Y}, f)$. Let Q be a separator of the policy space Π of size d. Let D be the uniform distribution on $[0, 1/\eta]$ for $\eta = \delta / \sqrt{2T(1 + 2T^{-1/2})(1+\delta)}$. Then Algorithm 2 applied to the contextual problem $(\mathcal{X}_c, \mathcal{Y}_c, f_c)$ with the contextual Q-extension $\mathbf{\Gamma}^Q$ of matrix $\mathbf{\Gamma}$ achieves the following regret:*

$$\textsc{Regret} \leq O(Nd\sqrt{T}/\delta).$$

Moreover, if there exists a contextual offline oracle $\textsc{Opt}(\cdot, \frac{1}{\sqrt{T}})$ which runs in time $\mathrm{poly}(N, d, M, T)$, then Algorithm 2 can be implemented efficiently in time $\mathrm{poly}(N, d, M, T)$.

Several examples of policy spaces with small separators, i.e., separators whose size is poly-logarithmic in the number of contexts, are presented by Goldman et al. [25] and Syrgkanis et al. [53]. In their examples, contexts are binary vectors and the task is binary classification; the examples include Boolean conjunctions, Boolean disjunctions, logarithmic-depth read-once majority formulas, and logarithmic-depth read-once positive NAND formulas. We next present an online auction design example where a Boolean disjunction is used to discriminate between two vectors of reserve prices.

Example 5.7 (Price Discrimination based on ORs of Binary Features). *We consider the class of single-item second price-auctions with discretized bidder-specific reserves, $\mathcal{I}_m$, where the arriving bidders are described by binary feature vectors in $\{0,1\}^K$, constituting the context σ. In Section 3.1, our goal was to identify a single vector of reserves $\mathbf{r} \in \mathcal{I}_m$ that would optimize the revenue. Here, we would like to achieve a higher revenue by splitting the market into two segments according to the contextual information and using a separate vector of reserves for each market segment. Formally, we define the set of policies $\Pi = \{\pi_{\mathbf{r}_0, \mathbf{r}_1, S} \mid \mathbf{r}_0 \in \mathcal{I}_m, \mathbf{r}_1 \in \mathcal{I}_m, S \subseteq [K]\}$, where $\mathbf{r}_0$ and $\mathbf{r}_1$ are the reserve vectors for*

the two respective segments and S determines the two segments as follows: $\Sigma_0 = \left\{\sigma \;\middle|\; \bigvee_{i\in S} \sigma_i = 0\right\}$, $\Sigma_1 = \left\{\sigma \;\middle|\; \bigvee_{i\in S} \sigma_i = 1\right\}$. *Thus,*

$$\pi_{\mathbf{r}_0,\mathbf{r}_1,S}(\sigma) = \begin{cases} \mathbf{r}_0 & \text{if } \sigma \in \Sigma_0 \\ \mathbf{r}_1 & \text{if } \sigma \in \Sigma_1. \end{cases}$$

Our contextual online learning algorithm over this policy space competes with a variety of market splitting strategies. For instance, we can do as well as an auctioneer that picks some high-reserve vector $\mathbf{r}_1$ and a low-reserve vector $\mathbf{r}_0$, learns which of the Boolean coordinates are indicative of a high valuation for the players, and when any one of them is 1, it uses the high reserves $\mathbf{r}_1$, and otherwise the low reserves $\mathbf{r}_0$.

To show that we can indeed efficiently compete with the class Π we just need to show that it has a small separator. We argue that the set $V = \{e_i \mid i \in [K]\} \cup \{\mathbf{1}, \mathbf{0}\}$ is a separator for Π. Consider any two policies $\pi_{\mathbf{r}_0,\mathbf{r}_1,S}$ and $\pi_{\mathbf{r}'_0,\mathbf{r}'_1,S'}$ that differ on some context σ. Then these policies must also differ on one of the following $\sigma' \in V$:

$$\sigma' = \begin{cases} \mathbf{1} & \text{if } \bigvee_{i\in S} \sigma_i = \bigvee_{i\in S'} \sigma_i = 1 \\ \mathbf{0} & \text{if } \bigvee_{i\in S} \sigma_i = \bigvee_{i\in S'} \sigma_i = 0 \\ \mathbf{e}_j, \text{ for } j \in S \setminus S' & \text{if } \bigvee_{i\in S} \sigma_i = 1, \bigvee_{i\in S'} \sigma_i = 0 \\ \mathbf{e}_j, \text{ for } j \in S' \setminus S & \text{if } \bigvee_{i\in S} \sigma_i = 0, \bigvee_{i\in S'} \sigma_i = 1. \end{cases}$$

Therefore, the size of the separator is $d = O(K)$, while the size of the policy space is exponential in K. Similar analysis can be done when the policies are using the AND of Boolean features rather than the OR.

5.2 Transductive Contextual Learning

We now consider the case where the learner knows a priori the set of contexts that could potentially arise, i.e., we assume that he knows that all contexts $\sigma_1, \ldots, \sigma_T$ come from a set P, which we will refer to as the *transductive set*. We do not require that the learner knows the multiplicity of each context or the sequence under which contexts arrive. Moreover, the set P could be of size as large as T, and in fact our analysis allows $|P| = o(T^2)$.

In this setting, ignoring some technical details, we can treat the transductive set P as a separator defined in the previous section. However, using the analysis of the previous section, we would obtain the regret guarantee that would grow linearly or super-linearly in T when the size of the set P is $\Omega(\sqrt{T})$. Thus in order to guarantee sub-linear regret we need a tighter analysis. To achieve this we will leverage the fact that in the transductive setting P is the whole set of contexts that will arise, which is a stronger property than being a separator. This will allow us to prove a stronger stability result than obtained via Lemma 2.4 in the previous section. Moreover, we will use a perturbation distribution D that has mean zero, leading to cancellations in the cumulative error term, which make that term not grow linearly with the size of P but rather as the square root of the size of P. The combination of these two improvements leads to sub-linear regret when $|P| = o(T^2)$.

We begin with a non-contextual learning problem $(\mathcal{X}, \mathcal{Y}, f)$ admitting a translation matrix $\mathbf{\Gamma} \in [0,1]^{|\mathcal{X}|\times N}$ that is implementable with complexity M and δ-admissible. We study the *transductive contextual problem* $(\mathcal{X}_c, \mathcal{Y}_c, f_c, P)$, i.e., the contextual problem $(\mathcal{X}_c, \mathcal{Y}_c, f_c)$ with the additional restriction that $\sigma_t \in P$, and analyze the regret of Algorithm 2 with the contextual extension $\mathbf{\Gamma}^P$ of matrix $\mathbf{\Gamma}$.

We first show an improved stability property of this algorithm. Even though the number of columns of matrix $\mathbf{\Gamma}^P$ is $|P| \cdot N$, we can show that the stability of the algorithm does not depend on $|P|$. The proof of the next lemma appears in Appendix E.3.

Lemma 5.8. *Let $\mathbf{\Gamma} \in [0,1]^{|\mathcal{X}|\times N}$ be δ-admissible for the non-contextual problem $(\mathcal{X}, \mathcal{Y}, f)$. Then the Generalized FTPL algorithm for the transductive contextual problem $(\mathcal{X}_c, \mathcal{Y}_c, f_c, P)$ with the contextual extension $\mathbf{\Gamma}^P$ of $\mathbf{\Gamma}$ as the translation matrix and with a $\left(\rho, \frac{1+2\epsilon}{\delta}\right)$-dispersed distribution D achieves the stability bound*

$$\mathbb{E}\left[\sum_{t=1}^{T} f_c(\pi_{t+1}, (\sigma_t, y_t)) - f_c(\pi_t, (\sigma_t, y_t))\right] \leq 2TN\rho(1+\delta^{-1}).$$

We next show that the error term from Equation (1), in our case,

$$\mathbb{E}\left[\boldsymbol{\alpha} \cdot (\mathbf{\Gamma}^P_{\pi_1} - \mathbf{\Gamma}^P_{\pi^*})\right] \quad , \tag{8}$$

does not grow linearly with the size of the transductive set $|P|$ since we allow $|P| = \Omega(T)$. We achieve this by using a mean-zero distribution D, rather than a non-negatively supported one. For such mean-zero distributions we can show that the error term grows as the square root of the number of columns of the contextual-extension matrix, rather than linearly. Combining the two improvements we get the following theorem (see Appendix E for the proof), which is a more refined version of Theorem 2.5.

Theorem 5.9. *Let $\mathbf{\Gamma} \in [0,1]^{|\mathcal{X}|\times N}$ be δ-admissible for the non-contextual problem $(\mathcal{X}, \mathcal{Y}, f)$ and D be the uniform distribution on $[-\nu, \nu]$. Then the regret of the Generalized FTPL algorithm for the transductive contextual problem $(\mathcal{X}_c, \mathcal{Y}_c, f_c, P)$ with translation matrix $\mathbf{\Gamma}^P$ can be bounded as*

$$\textsc{Regret} \leq \frac{TN(1+2\epsilon)(1+\delta)}{\nu\delta^2} + 2\nu\sqrt{2N|P|\ln|\Pi|} + \epsilon T.$$

For $\nu = \frac{\sqrt{T(1+2\epsilon)(1+\delta)}}{\delta}\left(\frac{N}{|P|\ln|\Pi|}\right)^{\frac{1}{4}}$ the above bound becomes $O\Big(N^{\frac{3}{4}}\left(|P|\ln|\Pi|\right)^{\frac{1}{4}}\frac{\sqrt{T(1+2\epsilon)(1+\delta)}}{\delta} + \epsilon T\Big)$. Setting $\epsilon = T^{-1/2}$ yields the regret $O\Big(T^{1/2}N^{\frac{3}{4}}(|P|\ln|\Pi|)^{\frac{1}{4}}/\delta\Big)$, which becomes $O\Big((NT)^{\frac{3}{4}}(\ln|\Pi|)^{\frac{1}{4}}/\delta\Big)$ when $|P| = O(T)$.

It remains to argue the oracle efficiency of the Generalized FTPL algorithm. However, the implementability condition in Section 2 only allows for non-negatively supported distributions D. If we want to allow arbitrary distributions over reals, our translation matrix must be able to implement negative weights α_j. This means we need to be able to simulate the negative of the difference between any two entries of a column by a dataset of adversary actions. This leads to the following additional condition on the translation matrix.

Definition 5.10. *A matrix $\mathbf{\Gamma}$ is* negatively implementable *with complexity M if for each $j \in [N]$ there exist a weighted dataset S_j^-, with $|S_j^-| \leq M$, such that*

$$\text{for all } x, x' \in \mathcal{X}: \quad -\left(\Gamma_{xj} - \Gamma_{x'j}\right) = \sum_{(w,y)\in S_j^-} w\Big(f(x,y) - f(x',y)\Big).$$

In other words, for each $j \in [N]$ there is $c_j \in \mathbb{R}$, such that for all $x \in \mathcal{X}$, $\Gamma_{xj} = c_j - \sum_{(w,y)\in S_j^-} w f(x,y)$. In this case, we say that weighted datasets S_j^-, $j \in [N]$, negatively implement $\mathbf{\Gamma}$ with complexity M.

Using a matrix $\mathbf{\Gamma}$ that is both implementable and negatively implementable gives rise to an oracle-efficient variant of the Generalized FTPL provided in Algorithm 3, which allows the distribution D to be over both positive and negative reals. Similar to Theorem 2.10, it is easy to see that the output of this algorithm is equivalent to the output of Generalized FTPL and therefore has the same regret guarantee (we omit this proof as it is identical to the proof of Theorem 2.10).

Algorithm 3: Oracle-Based Generalized FTPL with distributions over positive and negative reals

Input: datasets S_j and S_j^-, $j \in [N]$, that implement and negatively implement $\mathbf{\Gamma} \in [0,1]^{|\mathcal{X}|\times N}$,
distribution D over $\mathbb{R}$,
an offline oracle Opt.
Draw $\alpha_j \sim D$ independently for $j = 1,\dots,N$.
for $j = 1,\dots,N$ **do**
If $\alpha_j < 0$, let $Q_j \coloneqq \{(-\alpha_j w, y) : (w,y) \in S_j^-\}$ be the scaled version of S_j^-.
If $\alpha_j \geq 0$, let $Q_j \coloneqq \{(\ \alpha_j w, y) : (w,y) \in S_j\ \}$ be the scaled version of S_j.
end for
for $t = 1,\dots,T$ **do**
Set $S = \Big\{(1,y_1),\dots,(1,y_{t-1})\Big\} \cup \bigcup_{j\leq N} Q_j$.
Play $x_t = \textsc{Opt}(S, \frac{1}{\sqrt{T}})$.
Observe y_t and receive payoff $f(x_t, y_t)$.
end for

Theorem 5.11. *If $\mathbf{\Gamma}$ is implementable and negatively implementable with complexity M, then Algorithm 3 is an oracle-efficient implementation of Algorithm 1 with $\epsilon = 1/\sqrt{T}$ and has per-round complexity $O\Big(T + NM\Big)$.*

As an immediate corollary, we obtain that the existence of a polynomial-time offline contextual oracle implies the existence of a polynomial-time transductive contextual online learner with regret $O(T^{3/4})$, whenever we have access to an implementable, negatively implementable, and admissible matrix.

Corollary 5.12. *Assume that $\mathbf{\Gamma} \in [0,1]^{|\mathcal{X}|\times N}$ is implementable and negatively implementable with complexity M and δ-admissible for the non-contextual problem $(\mathcal{X}, \mathcal{Y}, f)$. Also, assume there exists an offline contextual oracle $\textsc{Opt}(\cdot, \frac{1}{\sqrt{T}})$ which runs in time $\mathrm{poly}(N, M, T)$. Let D be the uniform distribution as defined in Theorem 5.9, with $\epsilon = T^{-1/2}$. Then Algorithm 3, with D, applied to the transductive contextual problem $(\mathcal{X}_c, \mathcal{Y}_c, f_c, P)$ with $|P| \leq T$ using the translation matrix $\mathbf{\Gamma}^P$ runs in time $\mathrm{poly}(N, M, T)$ and achieves regret $O\Big((NT)^{\frac{3}{4}} (\ln|\Pi|)^{\frac{1}{4}}/\delta\Big)$.*

Observe that the proofs of implementability for VCG auctions with bidder-specific reserves (Lemma 3.3) and envy-free item-pricing auctions (Lemma 3.7) can be easily modified to show that the translation matrices for these applications are not only implementable, but also negatively implementable. Thus, using the results in this section, we immediately obtain no-regret oracle-efficient algorithms for these auctions in transductive contextual setting.

5.3 Extensions and Connections

Negative Implementability for Contextual Problems with Small Separators. We introduced negative implementability and Algorithm 3 for transductive learning, but the algorithm and the analysis also apply to general contextual problems with small separators whenever the translation matrix is both implementable and negative implementable (see Appendix E for the proof):

Corollary 5.13. *Assume that $\mathbf{\Gamma} \in [0,1]^{|\mathcal{X}|\times N}$ is implementable and negatively implementable with complexity M and δ-admissible for the non-contextual problem $(\mathcal{X}, \mathcal{Y}, f)$. Let Q be a separator of the policy space Π of size d. Let D be the uniform distribution on $[-v, v]$ for $v = \frac{\sqrt{T(1+2T^{-1/2})(1+\delta)}}{\delta} \left(\frac{Nd}{\ln|\Pi|}\right)^{\frac{1}{4}}$.*

Then Algorithm 3, with D, applied to the contextual problem $(\mathcal{X}_c, \mathcal{Y}_c, f_c)$ *with the contextual Q-extension* $\mathbf{\Gamma}^Q$ *of matrix* $\mathbf{\Gamma}$ *achieves the following regret:*

$$\textsc{Regret} \le O\Big((Nd)^{3/4}(\ln|\Pi|)^{1/4}\sqrt{T}/\delta\Big).$$

Moreover, if there exists a contextual offline oracle $\textsc{Opt}(\cdot, \frac{1}{\sqrt{T}})$ *which runs in time* $\mathrm{poly}(N, d, M, T)$*, then Algorithm 3 can be implemented efficiently in time* $\mathrm{poly}(N, d, M, T)$*.*

This result improves on Corollary 5.6 whenever $\ln|\Pi| = o(Nd)$. This was the case in Example 5.7, where $N = O(n \ln m)$, $d = O(K)$, so $Nd = O(Kn \ln m)$, whereas $\ln|\Pi| = O(K + n \ln m)$. In general we can only argue that $\ln|\Pi| \le d \ln|\mathcal{X}| \le Nd/\delta$, so in some cases Corollary 5.6 might be preferable.

Separators for Non-contextual Problems. In Sections 5.1 and 5.2, we showed how to turn a non-contextual translation matrix into a contextual translation matrix when we have access to a separator. It turns out that the concept of a separator is also instrumental in constructing non-contextual translation matrices when the payoff function takes values in $\{0,1\}$. Consider a non-contextual problem described by $(\tilde{\mathcal{X}}, \tilde{\mathcal{Y}}, \tilde{f})$, where $\tilde{f}$ only takes values in $\{0,1\}$. To derive a translation matrix that is admissible and implementable, we consider the class of policies $\pi_{\tilde{x}}$ that implement learner's actions $\tilde{x}$, by defining $\pi_{\tilde{x}}(\tilde{y}) = \tilde{f}(\tilde{x}, \tilde{y})$. Assume that the class of policies $\Pi = \{\pi_{\tilde{x}}\}_{\tilde{x}\in\tilde{\mathcal{X}}}$ has a separator $Q \subseteq \tilde{\mathcal{Y}}$. We can use it to construct a translation matrix $\tilde{\mathbf{\Gamma}}$, with rows indexed by $\tilde{x} \in \tilde{\mathcal{X}}$ and columns indexed by $\tilde{y} \in Q$ such that $\tilde{\Gamma}_{\tilde{x},\tilde{y}} = \tilde{f}(\tilde{x}, \tilde{y})$. From the definition of the separator, this matrix is 1-admissible for $(\tilde{\mathcal{X}}, \tilde{\mathcal{Y}}, \tilde{f})$, because each row uniquely identifies $\tilde{x}$ and the matrix is 0/1-valued. It is also implementable with complexity 1 using datasets $\{(1, \tilde{y})\}$. Thus, the size of the separator of Π determines the number of columns of the translation matrix for $(\tilde{\mathcal{X}}, \tilde{\mathcal{Y}}, \tilde{f})$.

Our notions of admissibility and implementability in a sense generalize the notion of a separator by allowing arbitrary real-valued payoffs (rather than just binary) and allowing implementation by weighted datasets (rather than just single examples). In our examples, we have leveraged both of these extensions to design oracle-efficient no-regret algorithms for various problem classes with real-valued losses.

Comparison with the algorithm of Syrgkanis et al. [53]. Our contextual learning results extend the algorithm and guarantees of Syrgkanis et al. [53], who consider contextual problems where the learner's actions are K-dimensional binary vectors $x \in \mathcal{X} \subseteq \{0,1\}^K$, the adversary in each step picks a payoff function $y \in \mathcal{Y} \subseteq \{\mathcal{X} \to [0,1]\}$, and the actual payoff corresponds to the evaluation of y at x, i.e., $f(x, y) = y(x)$. Syrgkanis et al. [53] assume that $\mathcal{Y}$ contains all linear functions. Their algorithm can be viewed as a variant of Algorithm 3 with the non-contextual translation matrix $\mathbf{\Gamma}$ of size $|\mathcal{X}| \times K$ whose columns correspond to the coordinates of x. This matrix is 1-admissible, and also implementable and negatively implementable with complexity 1. In our notation, $N = K$ and $\delta = 1$.

Syrgkanis et al. [53] consider an additional parameter $m = \max_{x\in\mathcal{X}}\|x\|_1$ and prove contextual regret bounds of $O\Big(m^{1/4}d^{3/4}\sqrt{KT\ln|\Pi|}\Big)$ and $O\Big(m^{1/4}d^{1/4}\sqrt{KT\ln|\Pi|}\Big)$, respectively, for the small-separator and transductive setting, where d denotes either the size of the separator Q or the size of the transductive set P. Our corresponding bounds, implied by Corollary 5.13 and Theorem 5.9, are $O\Big(d^{3/4}K^{3/4}\sqrt{T}(\ln|\Pi|)^{1/4}\Big)$ and $O\Big(d^{1/4}K^{3/4}\sqrt{T}(\ln|\Pi|)^{1/4}\Big)$. Thus, our bounds match or improve the prior bounds whenever $\ln|\Pi| = \Omega(K/m)$.[8] More importantly, our work extends the prior results to

[8]Given the knowledge of m, it is possible to use the probabilistic method to construct an admissible translation matrix $\mathbf{\Gamma}'$ with $N' = O\Big(m\log(1 + \frac{K}{m})\Big)$ and $\delta' = \frac{1}{m}$, implementable with complexity 1. The algorithm with this matrix matches or

generic actions (rather than just binary vectors) and allows implementation by weighted datasets (rather than just single examples).

6 APPROXIMATE ORACLES AND APPROXIMATE REGRET

The Oracle-Based Generalized FTPL algorithm requires an oracle that chooses an action whose payoff is within a small additive error of the best action's payoff. In this section, we extend our analysis of Oracle-Based Generalized FTPL to work with oracles that return an action whose payoff is only a constant-factor approximation of the best payoff. Formally, we consider the following oracles:

Definition 6.1 (Offline C-Approximation Oracle). *Given $C \le 1$, an* offline C-approximation oracle *for a payoff function f and a learner action set $\mathcal{X}$, denoted C-$\textsc{Opt}_{(f,\mathcal{X})}$, is any algorithm that receives as input a weighted set of adversary actions $S = \{(w_\ell, y_\ell)\}_{\ell \in \mathcal{L}}$, $w_\ell \in \mathbb{R}^+$, $y_\ell \in \mathcal{Y}$, and an accuracy parameter ϵ, and returns $x \in \mathcal{X}$, such that*

$$\sum_{(w,y)\in S} w f(x,y) \ge C \max_{x\in\mathcal{X}} \sum_{(w,y)\in S} w f(x,y) - \epsilon.$$

Similarly, we will write $\textsc{Opt}_{f,\mathcal{X}}$ for an offline oracle Opt defined in Section 2.2, making the dependence on f and $\mathcal{X}$ explicit. It corresponds to a C-approximation oracle with $C = 1$.

As discussed earlier, a minimal assumption for computationally efficient no-regret online learning is the existence of an efficient offline oracle $\textsc{Opt}_{f,\mathcal{X}}$ with a small additive error. When there exists a *fully polynomial-time approximation scheme* (FPTAS) for the offline optimization problem, i.e., an algorithm achieving $C = 1 - \xi$ for any $\xi > 0$, while running in time polynomial in the input size and $1/\xi$, then choosing $\xi = \epsilon/T$ recovers the additive sub-optimality for offline optimization and hence yields a no-regret guarantee for Oracle-Based Generalized FTPL. However, when the best polynomial-time approximation for a problem is a C-approximation for some constant $C < 1$, there is no hope for simultaneously achieving no-regret and computational efficiency [14, 50]. Instead, we can obtain performance guarantees for an alternative notion of regret, introduced by Kakade et al. [36], called C-Regret. Formally, the C-Regret of an online maximization problem is defined as

$$C\text{-}\textsc{Regret} = \mathbb{E}\left[C \max_{x\in\mathcal{X}} \sum_{t=1}^{T} f(x, y_t) - \sum_{t=1}^{T} f(x_t, y_t)\right].$$

Simply running Oracle-Based Generalized FTPL with an arbitrary C-approximation oracle for $C < 1$ does not automatically yield C-Regret $= o(T)$, but there are several important cases where this happens. Below we show that constant-factor approximation oracles obtained through relaxation of the objective and through Maximal-in-Range (MIR) algorithms yield sublinear C-Regret.

6.1 Approximation through Relaxation

A large class of approximation algorithms achieve their approximation guarantees by optimizing a relaxation of the payoff function. More formally, if there is a function $F : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$, such that $Cf(x,y) \le F(x,y) \le f(x,y)$ and there is an offline oracle $\textsc{Opt}_{F,\mathcal{X}}$, then it is clear that any online algorithm with sublinear regret for F also achieves a sublinear C-Regret for f. More formally:

Theorem 6.2. *Let F be a function such that for any $x \in \mathcal{X}$ and $y \in \mathcal{Y}$, $f(x,y) \ge F(x,y) \ge Cf(x,y)$. Let $\mathbf{\Gamma}^F \in [0,1]^{|\mathcal{X}|\times N}$ be δ-admissible and implementable with complexity M for the payoff function F. Then Algorithm 2 with distribution D as defined in Theorem 2.5, datasets that implement $\mathbf{\Gamma}^F$,*

improves the bounds of Syrgkanis et al. [53] if $\ln|\Pi| = \Omega\left(\frac{m^6}{K^2}\ln^3(1+\frac{K}{m})\right)$. The algorithm with $\mathbf{\Gamma}'$ also improves over our regret bounds for the matrix $\mathbf{\Gamma}$ if m is small; specifically, if $m = o\left((K/\ln K)^{3/7}\right)$.

and $\textsc{Opt}_{F,\mathcal{X}}$ *as an oracle is oracle-efficient with per-round complexity* $O(T + NM)$ *and* C-$\textsc{Regret} = O(N\sqrt{T}/\delta)$ *for* f.

A similar observation was made by Balcan and Blum [3] regarding approximation algorithms that use linear optimization as a relaxation and therefore can be efficiently optimized by the standard FTPL algorithm of Kalai and Vempala [37]. Our work extends this observation to any C-relaxation of the payoff function f that has an FPTAS and an admissible and implementable translation matrix.

Roughgarden and Wang [50] as a Relaxation. The approach of Roughgarden and Wang [50] for achieving a 1/2-regret for single-item second-price auctions with bidder-specific reserves falls exactly in the relaxation approximation framework. They give a relaxed objective which admits a polynomial-time offline oracle and which is always within a factor of two of the original objective. Then they run an oracle-based online learning algorithm for the relaxed objective. In their case, the relaxed objective yields a linear optimization problem and can be solved with the standard FTPL algorithm of Kalai and Vempala [37]. The theorem above shows that the same approach works even when the relaxed objective does not reduce to a linear optimization problem but to a problem that can be tackled by our Oracle-Based Generalized FTPL, providing a potential avenue for obtaining sublinear C-$\textsc{Regret}$ for values of $C \geq 1/2$.

Subsequent to the work of Roughgarden and Wang [50], Derakhshan et al. [15] designed an offline optimization algorithm that achieves a 0.684 fraction of the offline-optimal revenue. Their approach is based on an LP relaxation followed by rounding, but the style of their relaxation differs from the one required by Theorem 6.2, so our analysis does not apply.

6.2 Approximation by Maximal-in-Range Algorithms

Another interesting class of approximation algorithms to which we can apply our framework is Maximal-in-Range (MIR) algorithms. An MIR algorithm commits to a set of actions $\mathcal{X}' \subseteq \mathcal{X}$, independently of the input, and outputs the best $x \in \mathcal{X}'$. The set of actions $\mathcal{X}'$ has the property that an optimum over $\mathcal{X}'$ is a C-approximation of an optimum over $\mathcal{X}$. Thus, an MIR C-approximation algorithm forms an approximation oracle C-$\textsc{Opt}_{f,\mathcal{X}} := \textsc{Opt}_{f,\mathcal{X}'}$. Thanks to the C-approximate optimality of $\mathcal{X}'$, any online algorithm with sublinear regret relative to $\mathcal{X}'$ also achieves a sublinear C-$\textsc{Regret}$ relative to $\mathcal{X}$:

$\textsc{Theorem}$ 6.3. *Let* $\mathbf{\Gamma}^{\mathcal{X}'} \in [0,1]^{|\mathcal{X}'|\times N}$ *be* δ*-admissible and implementable with complexity* M *for the action set* $\mathcal{X}'$*. Then Algorithm 2 with distribution* D *as defined in Theorem 2.5, datasets that implement* $\mathbf{\Gamma}^{\mathcal{X}'}$*, and an MIR approximation oracle* $\textsc{Opt}_{f,\mathcal{X}'}$ *is oracle-efficient with per-round complexity* $O(T+NM)$ *and* C-$\textsc{Regret} = O(N\sqrt{T}/\delta)$ *relative to the action set* $\mathcal{X}$.

We will use Theorem 6.3 in Section 7.1 with a well-known Maximal-in-Range approximation algorithm for offline welfare maximization in multi-unit auctions to give a polynomial-time algorithm that achieves sublinear C-$\textsc{Regret}$ in the online setting.

7 ADDITIONAL APPLICATIONS

In this section, we work out two additional applications of our oracle-efficient online learning approach: online welfare maximization in multi-unit auctions and no-regret learning in simultaneous second-price auctions. For readability, we use a parenthesized superscript instead of a plain subscript to denote objects appearing in the round t, e.g., $v_i^{(t)}$.

7.1 Polynomial-Time Algorithm for Online Welfare Maximization in Multi-Unit Auctions

In this section, we consider the class of single-item multi-unit auctions that has a 1/2-approximation Maximal-in-Range (MIR) algorithm. We show how the results of Section 6 can be applied to this problem to achieve a *truly (oracle-free) polynomial-time* online algorithm with vanishing 1/2-Regret.

We consider an online learning variant of an n-bidder s-unit environment, where the goal is to allocate s identical items. Each bidder i has a monotone non-decreasing valuation function $v_i : \mathbb{N} \to [0,1]$, with $v_i(0) = 0$, describing the utility for receiving any given number of items. Equivalently, the bidder i has a non-negative *marginal valuation* $\mu_i(\ell) = v_i(\ell) - v_i(\ell-1)$ for receiving its ℓ^{th} item, and his total utility for receiving q_i items is $v_i(q_i) = \sum_{\ell=1}^{q_i} \mu_i(\ell)$. The goal is to find an allocation $\mathbf{q} \in \mathbb{N}^n$, subject to $\sum_{i=1}^n q_i = s$, that maximizes the total welfare $\sum_{i=1}^n v_i(q_i)$. Since we would like to allow settings where s is much larger than n, we seek algorithms running in time polynomial in n and independent of s.[9] In the online learning setting, every round t a fresh set of bidders arrive with new valuations $v_i^{(t)}$ and the learner commits to an allocation of the units of the item to the players, prior to seeing the valuations. The goal of the learner is to pick an allocation each day that competes with the best allocation in hindsight.

It is not hard to see that the offline welfare maximization problem in this setting corresponds to the Knapsack problem, where each player has a valuation equal to the average value in hindsight, i.e., $\frac{1}{T}\sum_{t=1}^T v_i^{(t)}(\cdot)$. Thus, dynamic programming can be used to compute a welfare-maximizing allocation in time polynomial in n and s. Dobzinski and Nisan [19] introduced a 1/2-approximation MIR algorithm for this problem. If $s \le n^2$, the algorithm uses the dynamic programming to directly maximizes the welfare (without any approximation). Otherwise, the algorithm divides s items into $\lfloor s/Q \rfloor$ bundles of size $Q := \lfloor s/n^2 \rfloor$, and one bundle of size $Q' < Q$ containing all the remaining items (if s in not a multiple of Q). Then, the MIR algorithm chooses the best allocation from the set of allocations (range of the algorithm) where all the items in one bundle are allocated to the same bidders. This is effectively a knapsack problem over $\lfloor s/Q \rfloor < 2n^2$ identical items and possibly one additional and distinct item, which can be solved in time polynomial in n.

We show how to construct a matrix $\mathbf{\Gamma}^{\text{MU}'}$ that is admissible and implementable for the allocations in the range of this MIR algorithm and then use Theorem 6.3 to obtain an online algorithm with vanishing 1/2-Regret, running in time $\text{poly}(n,T)$.

Construction of $\mathbf{\Gamma}$*:* Let $\mathcal{G}$ by the feasible set of the number of items assigned to any trader under allocations in the range of the MIR algorithm of Dobzinski and Nisan [19] described above, that is, $\mathcal{G} := \left\{aQ + bQ' : a \in \{0,1,\dots,\lfloor s/Q \rfloor\}, b \in \{0,1\}\right\}$. The case $s \le n^2$ is included by setting $Q = 1$ and $Q' = 0$. Denote the cardinality of $\mathcal{G}$ as r and let $g_0,\dots,g_{r-1}$ be the elements of $\mathcal{G}$ in the sorted order (note that $g_0 = 0$). Since $\lfloor s/Q \rfloor < 2n^2$, we have $r \le 4n^2$.

Let $\mathbf{\Gamma}^{\text{MU}'}$ be a matrix with n columns, such that for any allocation $\mathbf{q} = (g_{\iota_1},\dots,g_{\iota_n}) \in \mathcal{X}'$ and any column j, $\Gamma^{\text{MU}'}_{\mathbf{q},j} = \iota_j/r$. Clearly, for any $\mathbf{q} \ne \mathbf{q}' \in \mathcal{X}'$ we have $\mathbf{\Gamma}^{\text{MU}'}_{\mathbf{q}} \ne \mathbf{\Gamma}^{\text{MU}'}_{\mathbf{q}'}$, so $\mathbf{\Gamma}^{\text{MU}'}$ is $1/r$-admissible. Since $r \le 4n^2$, $\mathbf{\Gamma}^{\text{MU}'}$ is also $1/(4n^2)$-admissible.

It is not hard to see that $\mathbf{\Gamma}^{\text{MU}'}$ is also implementable with complexity 1. For any column j, consider the bid profile $\mathbf{v}^j$ where bidder j's marginal valuation is $1/r$ at the items reaching the allocation levels $g_\iota \in \mathcal{G}$ and other bidders have 0 valuation for any number of items. That is, $\mu_j(\ell) = \frac{1}{r}\mathbf{1}_{(\ell \in \mathcal{G})}$ and $\mu_i(\ell) = 0$ for all ℓ and $i \ne j$. The welfare of any allocation $\mathbf{q} = (g_{\iota_1},\dots,g_{\iota_n}) \in \mathcal{X}'$ on this bid profile is the utility of bidder j, which is $\iota_j/r = \Gamma^{\text{MU}'}_{\mathbf{q},j}$. Therefore, $\mathbf{\Gamma}^{\text{MU}'}$ is implementable with

[9] Recall that we assume that numerical computations are $O(1)$.

complexity 1 using datasets $S_j = \{(1, \mathbf{v}^j)\}$ for all $j \in [n]$. By Theorem 6.3 we therefore obtain the following:

Theorem 7.1. *Consider the problem of welfare maximization in s-unit auctions. Then Algorithm 2 with distribution D as defined in Theorem 2.5, datasets that implement $\mathbf{\Gamma}^{MU'}$, and the 1/2-approximate MIR algorithm of [19] as an oracle, runs in per-round time* $\text{poly}(n,T)$ *and plays the sequence of allocations $\mathbf{q}^{(1)}, \ldots, \mathbf{q}^{(T)}$, such that*

$$\frac{1}{2}\text{-Regret} = \mathbb{E}\left[\frac{1}{2}\left(\max_{\mathbf{q}\in\mathbb{N}^n:\ \|\mathbf{q}\|_1 = s} \sum_{t=1}^{T}\sum_{i=1}^{n} v_i^{(t)}(q_i)\right) - \sum_{t=1}^{T}\sum_{i=1}^{n} v_i^{(t)}(q_i^{(t)})\right] \leq O(n^4\sqrt{T}).$$

In the above theorem, the extra factor of n in the regret as compared to that implied by Theorem 6.3 is due to the fact that the maximum welfare in this problem is upper bounded by n.

7.2 Oracle-Efficient No-Regret Learning in Simultaneous Second-Price Auctions

In this section, we answer an open problem raised by Daskalakis and Syrgkanis [14] regarding the existence of an oracle-based no-regret algorithm for optimal bidding in Simultaneous Second-Price Auctions. We show that our Oracle-Based Generalized FTPL algorithm used with an appropriate implementable and admissible translation matrix can be used to obtain such an algorithm.

A *Simultaneous Second-Price Auction (SiSPA)* [4, 12, 23] is a mechanism for allocating k items to n bidders. Each bidder $i \leq n$ submits k simultaneous bids denoted by a vector of bids $\mathbf{b}_i$. The mechanism allocates each item using a second-price auction based on the bids solely submitted for this item, while breaking ties in favor of bidders with lower indices. For each item j, the winner is charged p_j, the second highest bid for that item (in the presence of ties, the numerical value of p_j may coincide with the value of the winning bid). Each bidder i has a fixed combinatorial valuation function $v_i : \{0,1\}^k \to [0,1]$ over bundles of items. Then, the total utility of bidder i who is allocated the bundle $\mathbf{q}_i \in \{0,1\}^k$ is $v_i(\mathbf{q}_i) - \mathbf{p}\cdot\mathbf{q}_i$, where $\mathbf{p}$ is the vector of second largest bids across all items.

We consider the problem of optimal bidding in a SiSPA from the perspective of the last bidder. Hereafter, we drop the indices of the bidders from the notation. From this perspective, the utility of the bidder only depends on its bid $\mathbf{b}$ and the threshold vector $\mathbf{p}$ of the second largest bids. The online optimal bidding problem is defined as follows.

Definition 7.2 (Online Bidding in SiSPAs [53]). *At each round t, the bidder picks a bid vector $\mathbf{b}^{(t)}$ and an adversary picks a threshold vector $\mathbf{p}^{(t)}$. The bidder wins the bundle of items $\mathbf{q}(\mathbf{b}^{(t)}, \mathbf{p}^{(t)})$, with $q_j(\mathbf{b}^{(t)}, \mathbf{p}^{(t)}) = \mathbf{1}_{(b_j^{(t)} > p_j^{(t)})}$ and gets the utility*

$$u(\mathbf{b}^{(t)}, \mathbf{p}^{(t)}) = v\left(\mathbf{q}(\mathbf{b}^{(t)}, \mathbf{p}^{(t)})\right) - \mathbf{p}^{(t)}\cdot\mathbf{q}(\mathbf{b}^{(t)}, \mathbf{p}^{(t)}).$$

We consider this problem under the *no-overbidding condition* that requires that for any bundle $\mathbf{q}$, the sum of bids over items in $\mathbf{q}$ does not exceed the bidder's valuation for $\mathbf{q}$, i.e., $\mathbf{b}^{(t)}\cdot\mathbf{q} \leq v(\mathbf{q})$, for all $\mathbf{q} \in \{0,1\}^k$. Similar no-overbidding assumptions are used in the previous work to prove that no-regret learning in second-price auctions has good welfare guarantees [12, 23].

We consider the online bidding problem where all the bids b_j are multiples of $1/m$ and the valuation function $v(\cdot)$ only takes values in $\{0, 1/m, \ldots, m/m\}$. We represent by $\mathcal{B}_m$ the class of all such bid vectors that satisfy the no-overbidding condition for $v(\cdot)$. The assumption on the valuation function is not restrictive, because any valuation function $v(\cdot)$ can be rounded down to the closest multiples of $1/m$ while losing at most $1/m$ utility, and the set $\mathcal{B}_m$ derived for the rounded-down valuation contains exactly the discretized bid vectors that satisfy no-overbidding condition for the

original valuation. Moreover, a similar discretization for the bid vectors was used by Daskalakis and Syrgkanis [53] for studying offline and online optimal bidding in SiSPAs.

Next, we show how to construct an implementable and admissible translation matrix for $\mathcal{B}_m$.

Construction of $\mathbf{\Gamma}$*:* Let $\mathbf{\Gamma}^{\mathrm{OB}}$ be a matrix with k columns and $|\mathcal{B}_m|$ rows that are equal to bid vectors, i.e., $\mathbf{\Gamma}^{\mathrm{OB}}_{\mathbf{b}} = \mathbf{b}$.

The next lemma shows that $\mathbf{\Gamma}^{\mathrm{OB}}$ is admissible and implementable.

LEMMA 7.3. *$\mathbf{\Gamma}^{OB}$ is $1/m$-admissible and implementable with complexity m.*

PROOF. Since $\mathbf{\Gamma}^{\mathrm{OB}}_{\mathbf{b}} = \mathbf{b}$ and $\mathbf{b}$ is discretized, $\mathbf{\Gamma}^{\mathrm{OB}}$ is $1/m$-admissible. Next, we show that the jth column of $\mathbf{\Gamma}^{\mathrm{OB}}$ can be implemented by a weighted dataset S_j containing m threshold vectors where all but the jth threshold are set to 1. Specifically, for $\ell = 0, 1, \ldots, m-1$, let $\mathbf{p}_\ell = (\ell/m)\mathbf{e}_j + \sum_{j' \neq j} \mathbf{e}_{j'}$. Note that the utility of playing a bid $\mathbf{b}$ against $\mathbf{p}_\ell$ is $u(\mathbf{b}, \mathbf{p}_\ell) = \Big(v(\mathbf{e}_j) - \ell/m\Big)\mathbf{1}_{(b_j > \ell/m)}$. We set the weight corresponding to $\mathbf{p}_\ell$ to

$$w_\ell = \begin{cases} \frac{1}{m} \cdot \frac{1}{v(\mathbf{e}_j) - \ell/m} & \text{if } \ell/m < v(\mathbf{e}_j), \\ 0 & \text{otherwise.} \end{cases}$$

Since $b_j \leq v(\mathbf{e}_j)$ for any $\mathbf{b}$, we have

$$\begin{aligned} \sum_{\ell=0}^{m-1} w_\ell \, u(\mathbf{b}, \mathbf{p}_\ell) &= \sum_{\ell=0}^{m-1} \frac{1}{m} \cdot \frac{1}{v(\mathbf{e}_j) - \ell/m} \cdot \Big(v(\mathbf{e}_j) - \ell/m\Big)\mathbf{1}_{(b_j > \ell/m)} \\ &= \sum_{\ell=0}^{m-1} \frac{1}{m} \mathbf{1}_{(b_j > \ell/m)} = \sum_{\ell=0}^{mb_j - 1} \frac{1}{m} \\ &= b_j = \Gamma^{\mathrm{OB}}_{\mathbf{b},j}. \end{aligned}$$

Thus, indeed $S_j = \{(w_\ell, \mathbf{p}_\ell)\}_\ell$ implements $\mathbf{\Gamma}^{\mathrm{OB}}$. Note that $|S_j| = m$, so $\mathbf{\Gamma}^{\mathrm{OB}}$ is implementable with complexity m. □

The next theorem is a direct consequence of Lemma 7.3 and Theorems 2.5 and 2.10.

THEOREM 7.4. *Consider the problem of Online Bidding in SiSPAs. Let D be the uniform distribution as described in Theorem 2.5. Then, the Oracle-Based Generalized FTPL algorithm with D and datasets that implement $\mathbf{\Gamma}^{\mathrm{OB}}$ is oracle-efficient with per-round complexity $O(T + km)$ and has regret*

$$\mathbb{E}\left[\max_{\mathbf{b} \in \mathcal{B}_m} \sum_{t=1}^{T} u(\mathbf{b}, \mathbf{p}^{(t)}) - \sum_{t=1}^{T} u(\mathbf{b}^{(t)}, \mathbf{p}^{(t)})\right] \leq O(km\sqrt{T}).$$

8 CONCLUSION

In this paper, we have studied oracle-efficient online learning algorithms and applied them to online auction design. Several previous works have also studied oracle-efficient online learning, including Kalai and Vempala [37], whose Follow-the-Perturbed-Leader algorithm is oracle-efficient when the payoff is linear, and Daskalakis and Syrgkanis [14], whose algorithm is oracle-efficient when the adversary's action set is of polynomial size. On the other hand, Hazan and Koren [30] have shown that oracle-efficient online learning is not always achievable. In this paper, we have advanced the study of conditions under which oracle-efficient online learning is possible and introduced a general-purpose oracle-efficient algorithm that works under weaker structural assumptions than previous approaches. We also showed that many problems in auction design demonstrate such a structure. As a result, our framework gave oracle-efficient online algorithms for a large number of

problems in auction design when the environment, e.g., bidders' valuations, changes adversarially over a period of time.

Many open problems remain. First and more broadly, our work provides sufficient conditions for existence of oracle-efficient online learning algorithms. Since oracle-efficient algorithms do not always exist, a natural direction is to extend the scope of sufficient conditions and also study necessary conditions for their existence. Second, our oracle-efficient algorithm obtains a regret of $O\left(N\sqrt{T}/\delta\right)$ when there is a δ-admissible and implementable translation matrix with N columns. Forgoing implementability, any 0/1-valued payoff function has an admissible translation matrix with $N = \log|\mathcal{X}|$ columns, so the Generalized FTPL algorithm can always achieve the regret $O(\sqrt{T}\log|\mathcal{X}|)$, albeit not efficiently. However, this means that there is a gap of $\sqrt{\log|\mathcal{X}|}$ between the information-theoretically optimal regret bound, i.e., $O(\sqrt{T\log|\mathcal{X}|})$, and the one obtained by our algorithm. Are there oracle-efficient algorithms that obtain the information-theoretically optimal regret bounds for problems that have admissible and implementable translation matrices? Answering these questions will enable a deeper understanding of the inherent tradeoffs between the computational and information-theoretic aspects of online learning.

A FURTHER RELATED WORK

Online Learning. Study of online no-regret algorithms goes back to the seminal work of Hannan [27], who was the first to develop algorithms with regret $\text{poly}(|\mathcal{X}|)o(T)$. Motivated by settings in machine learning, game theory, and optimization, algorithms with regret $\log(|\mathcal{X}|)o(T)$ were later developed by Littlestone and Warmuth [42] and Freund and Schapire [24]. From the computational perspective, however, algorithms with runtime that has sub-linear dependence on $|\mathcal{X}|$ remained elusive. Hazan and Koren [30] recently showed such algorithms cannot exist without additional assumptions on the structure of the optimization task. Our work introduces general structural conditions under which online oracle-efficient learning is possible and provides oracle-efficient no-regret results in this space.

For the case of linear objective functions, Kalai and Vempala [37] proposed the first oracle-efficient online algorithm. Their algorithm added perturbations that are independent across dimensions to the historical cumulative payoff of each action of the learner and then picked the action with the largest perturbed historical payoff. However, for non-linear functions, this algorithm runs in time $\text{poly}(|\mathcal{X}|)$.

Daskalakis and Syrgkanis [14] proposed an approach of augmenting the history of an online algorithm with additional "fake" historical samples of the adversary actions. In particular, the authors show that when the set of adversary's actions, $\mathcal{Y}$, is small, one can add a random number of copies of each of the actions of the adversary to the history and obtain a regret of $O(|\mathcal{Y}|\sqrt{T})$ with oracle-efficient per-round complexity of $\text{poly}(|\mathcal{Y}|, T)$. However, the adversary's set of actions is often large in many applications, e.g., applications to auction design, where this approach does not lead to a no-regret oracle-efficient algorithm.

Our work fills in this gap, by generalizing and extending the algorithm of Daskalakis and Syrgkanis [14] with arbitrary set of actions of the adversary that meet the admissibility requirement. Additionally, our work introduces what can be thought of as a measure of statistical and computational complexity of the problem that takes into account the effective size of a decomposition of the set of learner's actions, $\mathcal{X}$, on the actions of the adversary, $\mathcal{Y}$, not just the size of the learner's set of actions, as in [24, 37, 42], or the adversary's set of actions, as in [14].

Prior to our work, stylized algorithms with improved regret bounds or oracle-efficient learning have existed in multiple domains. Hazan et al. [32] study the regret bound of the problem of learning from experts, when the losses of experts are restricted to a low dimensional subspace. They show

that one can obtain a regret bound that only depends on the rank of this space, rather than the number of experts. In another work, Hazan and Kale [29] study online submodular minimization. They show that it is not necessary to add a perturbation to each action, but it suffices to add a perturbation on each underlying element of the set. This is a special case of our approach where $\mathbf{\Gamma}$ is implemented by datasets that include single-element sets.

Contextual learning in the transductive setting using an optimization oracle was previously studied by Kakade and Kalai [35], whose work was later extended and improved by Cesa-Bianchi and Shamir [10]. Syrgkanis et al. [53] further extended the results to more general settings (including the bandit setting) based on the concept of small separator, which appears in prior work under the name *universal identification sequence* [25]. Our definitions of admissibility and implementability can be viewed as non-contextual generalizations of the concept of the separator (see Section 5.3). We also show how to use the separator framework of Syrgkanis et al. [53] to extend our algorithm to contextual settings. Contextual bandits with an oracle have also been heavily studied in recent years [1, 21, 40, 48, 54].

Auction Design. The seminal work of Myerson [44] gave a recipe for designing the optimal truthful auction when the distribution over bidder valuations is completely known to the auctioneer. More recently, a series of works have focused on what can be done if the the distribution is not fully known to the auctioneer, but the auctioneer has access to some historical data. These approaches have focused on the sample complexity [13, 43] and computationally efficiency [17] of designing optimal auctions from sample valuations. That is, they answer how much data is needed for one to design a near optimal auction for a distribution. They use tools from supervised learning, such as pseudo-dimension [47] (a variant of VC dimension for real-valued functions) and compression bounds [41]. These tools are specially appropriate for when historical data in form of i.i.d samples from a distribution is available to the learner and their use does not extend to the adversarial online learning setting which we study. Our work drops any distributional assumptions and considers learning the best auction among a class of auctions in the adversarial setting. However, we show that our algorithms can be used to obtain sample complexity results for more general stochastic settings, such as fast mixing markov processes.

On a different front, a series of works have considered online learning in auctions and pricing mechanisms, when the size of the auction class is itself small. Variations of this problem have been studied by Blum and Hartline [5], Bubeck et al. [7], Cesa-Bianchi et al. [9], Kleinberg and Leighton [39]. In this case, standard no-regret algorithms (e.g., [24, 37]) obtain some no-regret results in polynomial time. However, additional effort is needed to design stylized algorithms for individual auction classes that obtain optimal regret, possibly even when the learner receives limited historical feedback. Our approach on the other hand works with classes of auctions that have exponentially large cardinality, thereby allowing one to use more expressive auctions to gain higher revenue.

More recently, Roughgarden and Wang [50] studied online learning in the class of single-item second-price auctions with bidder-specific reserves. The authors introduce an algorithm with performance that approaches a constant factor of the optimal revenue in hindsight. Our work goes beyond this auction class and introduces a general adversarial framework for auction design.

B OMITTED PROOFS FROM SECTION 2

B.1 Proof of Lemma 2.1

We first prove an approximate variant of Be-the-Leader Lemma.

Lemma B.1 (Be-the-Approximate-Leader Lemma). *In the Generalized FTPL algorithm, for any* $x \in \mathcal{X}$,

$$\sum_{t=1}^{T} f(x_{t+1}, y_t) + \boldsymbol{\alpha} \cdot \boldsymbol{\Gamma}_{x_1} \geq \sum_{t=1}^{T} f(x, y_t) + \boldsymbol{\alpha} \cdot \boldsymbol{\Gamma}_x - \epsilon(T+1) \ .$$

Proof. For $T = 0$ the inequality holds, because by ϵ-optimality of x_1, we have $\boldsymbol{\alpha} \cdot \boldsymbol{\Gamma}_{x_1} \geq \boldsymbol{\alpha} \cdot \boldsymbol{\Gamma}_x - \epsilon$ for all x. Assume that the claim holds for some T. Then, for all x,

$$\begin{aligned}
\sum_{t=1}^{T+1} f(x_{t+1}, y_t) + \boldsymbol{\alpha} \cdot \boldsymbol{\Gamma}_{x_1} &= \sum_{t=1}^{T} f(x_{t+1}, y_t) + \boldsymbol{\alpha} \cdot \boldsymbol{\Gamma}_{x_1} + f(x_{T+2}, y_{T+1}) \\
&\geq \sum_{t=1}^{T} f(x_{T+2}, y_t) + \boldsymbol{\alpha} \cdot \boldsymbol{\Gamma}_{x_{T+2}} - \epsilon(T+1) + f(x_{T+2}, y_{T+1}) \\
&\qquad \text{(by induction hypothesis)} \\
&= \sum_{t=1}^{T+1} f(x_{T+2}, y_t) + \boldsymbol{\alpha} \cdot \boldsymbol{\Gamma}_{x_{T+2}} - \epsilon(T+1) \\
&\geq \sum_{t=1}^{T+1} f(x, y_t) + \boldsymbol{\alpha} \cdot \boldsymbol{\Gamma}_x - \epsilon(T+2) \ , \qquad \text{(by } \epsilon\text{-optimality of } x_{T+2})
\end{aligned}$$

proving the lemma. □

Proof of Lemma 2.1. Let $x^* = \arg\max_{x \in \mathcal{X}} \sum_{t=1}^{T} f(x, y_t)$. Then by Lemma B.1,

$$\begin{aligned}
\text{Regret} &= \mathbb{E}\left[\sum_{t=1}^{T} f(x^*, y_t) - \sum_{t=1}^{T} f(x_t, y_t)\right] \\
&= \mathbb{E}\left[\sum_{t=1}^{T} f(x^*, y_t) - \sum_{t=1}^{T} f(x_{t+1}, y_t)\right] + \mathbb{E}\left[\sum_{t=1}^{T} f(x_{t+1}, y_t) - \sum_{t=1}^{T} f(x_t, y_t)\right] \\
&\leq \mathbb{E}\left[\boldsymbol{\alpha} \cdot (\boldsymbol{\Gamma}_{x_1} - \boldsymbol{\Gamma}_{x^*})\right] + \mathbb{E}\left[\sum_{t=1}^{T} f(x_{t+1}, y_t) - \sum_{t=1}^{T} f(x_t, y_t)\right] + \epsilon(T+1) \ .
\end{aligned}$$

□

C OMITTED PROOFS FROM SECTION 3

C.1 Proof of Equation (4)

Consider any vector of reserves $\mathbf{r} \in \mathcal{I}$ and let $\mathbf{r}' \in \mathcal{I}_m$ be the vector obtained by rounding each reserve price down to the nearest multiple of $1/m$, except for reserves that lie in the range $[0, 1/m)$, which are rounded up to $1/m$.

First we show that by increasing all the reserves in $\mathbf{r}$ that are below $1/m$ to the value $1/m$, we cannot decrease the revenue by more than $(|W| - |W''|)/m$ where W is the set of winners under $\mathbf{r}$ and W'' is the set of winners under the new reserves. Let $\mathbf{r}''$ be the vector of new reserves and let p_i and p_i'' be the payments of bidder i in auctions $\mathbf{r}$ and $\mathbf{r}''$. Let v_i be the valuation of bidder i. Let b and b'' denote the highest unserviced bids that cleared their reserve under $\mathbf{r}$ and $\mathbf{r}''$, respectively, or 0 if all the bidders that cleared their reserve were serviced. Recall that W and W'' are the sets of winners under $\mathbf{r}$ and $\mathbf{r}''$, respectively. First, note that the set of bidders that clear the reserves under $\mathbf{r}''$ is always a subset of those that clear the reserves under $\mathbf{r}$. However, if $b \geq 1/m$ then all the bids in W must be at least $1/m$, and so all the bidders in W will clear their reserves under $\mathbf{r}''$, and therefore $W'' = W$. Moreover, also the highest unserviced bidder under $\mathbf{r}$ clears the new reserve, and so $b'' = b$. Thus, for all the winners i, $p_i'' = \max\{r_i'', b''\} = \max\{r_i, 1/m, b\} = \max\{r_i, b\} = p_i$,

so the revenue under new reserves is the same as under old reserves. Now consider the case where $b < \frac{1}{m}$. This means that all the unserviced bids that cleared the reserves $\mathbf{r}$ are lower than $\frac{1}{m}$, and so they do not clear the new reserves $\mathbf{r}''$. As a result, the set of bidders that clear the new reserves is a subset of W, so $b'' = 0$ and $W'' \subseteq W$. In fact, W'' in this case consists exactly of the bidders that clear the new reserves. For a bidder $i \in W''$ we have

$$p_i'' = \max\{r_i'', b''\} = \max\{r_i, 1/m\} \geq \max\{r_i, b\} = p_i,$$

so the revenue obtained from $i \in W''$ can only increase under $\mathbf{r}''$ compared with $\mathbf{r}$. The bidders $i \in W \backslash W''$, that is those who win an item under $\mathbf{r}$ but not under $\mathbf{r}'$, must have a bid (and therefore a payment) smaller than $1/m$. Therefore, the overall revenue decreases by at most $(|W| - |W''|)/m$.

Now we assume that we start with a reserve price $\mathbf{r}$, such that $r_i \geq 1/m$ for all i, and let $\mathbf{r}'$ be the reserve vector where all reserves in $\mathbf{r}$ are rounded down to the nearest multiple of $1/m$. If $v_i > r_i$, then $v_i > r_i'$, so any bidder who would have been included in the basic VCG auction using reserves $\mathbf{r}$ is still included with $\mathbf{r}'$. This can only increase the number of bidders who are serviced and therefore pay a charge. We now show also that the total decrease in payment from bidders with value at least $1/m$ is at most s/m.

Consider the amount that serviced bidder i is charged. This is the maximum of r_i and the highest bid of a bidder in the basic VCG auction who was not serviced (or 0 if all bidders were serviced); let b denote this highest unserviced bid in the basic VCG auction under $\mathbf{r}$, and similarly let b' denote such a bid under $\mathbf{r}'$. Since the set of bidders entering the basic VCG auction increases from $\mathbf{r}$ to $\mathbf{r}'$, we must have $b' \geq b$.

Let U be the set of bidders serviced under $\mathbf{r}$, and U' the set under $\mathbf{r}'$. The difference in revenue is

$$\begin{aligned}\sum_{i \in U} \max\{r_i, b\} - \sum_{i \in U'} \max\{r_i', b'\} \\ = \sum_{i \in U \cap U'} (\max\{r_i, b\} - \max\{r_i', b'\}) + \sum_{i \in U \backslash U'} \max\{r_i, b\} - \sum_{i' \in U' \backslash U} \max\{r_{i'}', b'\}. \end{aligned} \tag{9}$$

We begin by analyzing the last two terms. For any $i \in U \setminus U'$ and $i' \in U' \setminus U$,

$$r_{i'}' + 1/m > r_{i'} > v_{i'} \geq v_i \geq r_i,$$

where $r_{i'} > v_{i'}$, because i' did not enter the basic VCG auction for $\mathbf{r}$ (otherwise it would have won), and $v_{i'} \geq v_i$, because i enters the basic VCG auction for $\mathbf{r}'$, but is not allocated the item. Therefore,

$$\max\{r_{i'}', b'\} \geq \max\left\{r_i - 1/m, b\right\} \geq \max\{r_i, b\} - 1/m.$$

Since $|U \setminus U'| \leq |U' \setminus U|$, we can pick $V \subseteq U' \setminus U$ such that $|V| = |U \setminus U'|$ and obtain

$$\sum_{i \in U \backslash U'} \max\{r_i, b\} - \sum_{i' \in U' \backslash U} \max\{r_{i'}', b'\} \leq \sum_{i \in U \backslash U'} \max\{r_i, b\} - \sum_{i' \in V} \max\{r_{i'}', b'\} \leq \frac{|U \setminus U'|}{m} .$$

Note that each term in the first sum of Equation (9) is at most $1/m$, because

$$\max\{r_i', b'\} \geq \max\left\{r_i - 1/m, b\right\} \geq \max\{r_i, b\} - 1/m.$$

Thus, we have

$$\mathrm{Rev}(\mathbf{r}, \mathbf{v}) - \mathrm{Rev}(\mathbf{r}', \mathbf{v}) \leq \frac{|U \cap U'|}{m} + \frac{|U \setminus U'|}{m} = \frac{|U|}{m} .$$

Thus if we start from any reserve price vector $\mathbf{r}$, we can first round up to $1/m$ all reserves that are below $1/m$ and then round down any other reserve to the nearest multiple of $1/m$. Denoting the initial set of winners W, the intermediate set of winners U and the final set of winners U', we have

shown that the revenue can drop by at most $(|W| - |U|)/m$ in the first step and by at most $|U|/m$ in the second step, so the overall drop is at most $|W|/m \leq s/m$. This yields the approximation result

$$\max_{\mathbf{r}\in\mathcal{I}} \sum_{t=1}^{T} \mathrm{Rev}(\mathbf{r}, \mathbf{v}_t) - \max_{\mathbf{r}\in\mathcal{I}_m} \sum_{t=1}^{T} \mathrm{Rev}(\mathbf{r}, \mathbf{v}_t) \leq \frac{Ts}{m}. \qquad \square$$

C.2 Proof of Lemma 3.7

We will describe an isomorphism with the setting in the proof of Lemma 3.3, which will allow us to directly apply the analysis of $\mathbf{\Gamma}^{\mathrm{VCG}}$ to $\mathbf{\Gamma}^{\mathrm{IP}}$. The isomorphism from the IP setting to VCG setting maps items ℓ in IP to bidders i in VCG, and price vectors $\mathbf{a}$ to reserve price vectors $\mathbf{r}$. We therefore assume that n in VCG equals k in IP, and the values of m in both settings are equal. Then, indeed $\mathbf{\Gamma}^{\mathrm{VCG}}$ equals $\mathbf{\Gamma}^{\mathrm{IP}}$.

Next we need to show how to construct S_j for all j in the $\mathbf{\Gamma}^{\mathrm{IP}}$ setting. Assume that j corresponds to the bidder i and the bit β in VCG setting, and the item ℓ and the bit β in IP setting. In VCG, we considered the bid profiles $\mathbf{v}_h = (h/m)\mathbf{e}_i$, and the revenue of any auction $\mathbf{r}$ is

$$\mathrm{Rev}^{\mathrm{VCG}}(\mathbf{r}, \mathbf{v}_h) = r_i \mathbf{1}_{(h \geq m r_i)}.$$

In IP setting, we consider profiles $\mathbf{v}'_h$ of combinatorial valuations over bundles $\mathbf{q} \in \{0,1\}^k$, in which all bidders have values zero on all bundles and one bidder has value h/m for bundle $\mathbf{e}_\ell$ and zero on all other bundles.[10] In this case, we have

$$\mathrm{Rev}^{\mathrm{IP}}(\mathbf{a}, \mathbf{v}'_h) = a_i \mathbf{1}_{(h \geq m a_i)}.$$

Thus, both the translation matrices $\mathbf{\Gamma}^{\mathrm{VCG}}$ and $\mathbf{\Gamma}^{\mathrm{IP}}$ as well as the revenue functions $\mathrm{Rev}^{\mathrm{VCG}}$ and $\mathrm{Rev}^{\mathrm{IP}}$ are isomorphic (given these choices of the profiles). Therefore, we can set the weights w'_h in IP setting equal to the weights w_h in VCG setting and obtain admissibility and implementability with the same constants and complexity. $\square$

C.3 Proof of Lemma 3.9

Single-minded setting. In the single-minded setting, each bidder i is interested in one particular bundle of items $\hat{\mathbf{q}}_i$. That is, $v_i(\mathbf{q}_i) = v_i(\hat{\mathbf{q}}_i)$ for all $\mathbf{q}_i \supseteq \hat{\mathbf{q}}_i$ and 0 otherwise. Consider any vector of prices $\mathbf{a} \in \mathcal{P}$ and let $\mathbf{a}' \in \mathcal{P}_m$ be the vector obtained by rounding up all prices that are below k/m to the nearest multiple of $1/m$ and reducing all prices that are at least k/m by $(k-1)/m$ and then rounding down to the nearest multiple of $1/m$. More formally,

$$a'_\ell = \begin{cases} \lceil a_\ell \rceil_{\frac{1}{m}} & \text{if } a_\ell < \frac{k}{m}, \\ \left\lfloor a_\ell - \frac{k-1}{m} \right\rfloor_{\frac{1}{m}} & \text{if } a_\ell \geq \frac{k}{m}, \end{cases}$$

where $\lceil \cdot \rceil_\epsilon$ and $\lfloor \cdot \rfloor_\epsilon$ indicate rounding up and down to the nearest multiple of ϵ, respectively. Note that $a'_\ell \in \{1/m, \ldots, (m-1)/m, 1\}$ by definition. Since $a_\ell \geq 0$ for all ℓ, we can assume without loss of generality that bidder i receives either the bundle $\mathbf{0}$ or $\hat{\mathbf{q}}_i$. Only the bidders that receive $\hat{\mathbf{q}}_i$ contribute towards the revenue under $\mathbf{a}$. For each such bidder i, there are two cases:

(1) All items ℓ in bundle $\hat{\mathbf{q}}_i$ are such that $a_\ell < k/m$. Since there are at most k items included in $\hat{\mathbf{q}}_i$, we have that $\hat{\mathbf{q}}_i \cdot \mathbf{a} < k^2/m$, i.e., the original payment for $\hat{\mathbf{q}}_i$ is at most k^2/m. Since the payment of the bidder i under $\mathbf{a}'$ is non-negative, regardless of the bundle it receives, the drop in the payment amount by the bidder i is at most k^2/m.

[10] Note that a simple variation of this bid profile can be used in settings where the valuations need to satisfy additional assumptions, such as (sub-)additivity or free disposal. In such cases, we can use a similar bid profile where one bidder has valuation h/m for any bundle that includes item ℓ and all other valuations are 0.

(2) There is an item j in bundle $\hat{\mathbf{q}}_i$ such that $a_j \geq k/m$. This means that there can be at most $k-1$ items whose original price was less than k/m, and each of these prices has increased by at most $1/m$. These items increase the price of the bundle $\hat{\mathbf{q}}_i$ by at most $(k-1)/m$. Since $a'_j \leq a_j - (k-1)/m$, item j decreases the price of the bundle by at least $(k-1)/m$. Altogether, we therefore have that $\hat{\mathbf{q}}_i \cdot \mathbf{a}' \leq \hat{\mathbf{q}}_i \cdot \mathbf{a}$, so the bundle $\hat{\mathbf{q}}_i$ is assigned to the bidder i under $\mathbf{a}'$. We also have that $a'_\ell \geq a_\ell - k/m$ for all ℓ, and since there are only k items, the price of the bundle $\hat{\mathbf{q}}_i$ drops by at most k^2/m.

Thus, in both cases, the payment of bidder i for the bundle it receives under $\mathbf{a}'$ is lower by at most k^2/m than the payment under $\mathbf{a}$. Therefore, for any valuation profile of single-minded bidders $\mathbf{v}$,

$$\mathrm{Rev}(\mathbf{a},\mathbf{v}) - \mathrm{Rev}(\mathbf{a}',\mathbf{v}) \leq \frac{nk^2}{m}.$$

Unit-demand setting. In the unit-demand setting with infinite supply, each bidder i has $v_i(\mathbf{e}_\ell)$ for item ℓ, and wishes to purchase *at most one item*, i.e., item $\arg\max_\ell (v_i(\mathbf{e}_\ell) - a_\ell)$. We show that for any $\mathbf{a} \in \mathcal{P}$ there is $\mathbf{a}'' \in \mathcal{P}_m$ such that for any valuation profile $\mathbf{v}$, $\mathrm{Rev}(\mathbf{a},\mathbf{v}) - \mathrm{Rev}(\mathbf{a}'',\mathbf{v}) \leq O(nk/m)$. At a high level, we first choose $\mathbf{a}'$, with $a'_\ell \in \{0, 1/m, \cdots, m/m\}$, as discounted prices such that items with higher prices are discounted by larger amounts. It is not hard to see that under this condition, no bidder purchases a less expensive item in auction $\mathbf{a}'$. So, the loss in the revenue of the auction is bounded by the discount on the items. Using this intuition, it is sufficient to find $\mathbf{a}'$, with $a'_\ell \in \{0, 1/m, \cdots, m/m\}$, such that $a_\ell \geq a'_\ell \geq a_\ell - O(k/m)$ for all $\ell \leq k$, and $a_\ell - a'_\ell \geq a_{\ell'} - a'_{\ell'}$ when $a_\ell \geq a_{\ell'}$. We then show how to derive $\mathbf{a}'' \in \mathcal{P}_m$ whose revenue is at least as good as $\mathbf{a}'$.

Without loss of generality, assume $a_1 \leq a_2 \leq \cdots \leq a_k$. For ease of exposition, let $\epsilon = 1/m$. To begin, let a'_1 be the largest multiple of ϵ less than or equal to a_1. For $\ell = 2, \ldots, k$, let a'_ℓ be the largest multiple of ϵ less than or equal to a_ℓ such that $a_\ell - a'_\ell \geq a_{\ell-1} - a'_{\ell-1}$. Note that a'_ℓ is well defined, since we can begin by considering $a'_\ell = a'_{\ell-1}$ and then increase by ϵ until the condition is violated. This construction means that pricing of items with a larger ℓ is more attractive in $\mathbf{a}'$ than it was in $\mathbf{a}$. Thus, no bidder that prefers an item ℓ under $\mathbf{a}$, would prefer any item $\ell' < \ell$ under $\mathbf{a}'$. Also, $a'_1 \leq a'_2 \leq \cdots \leq a'_k$. Therefore, the revenue obtained from the bidder i who prefers ℓ under $\mathbf{a}$ is at least a'_ℓ under $\mathbf{a}'$, which implies the bound

$$\mathrm{Rev}(\mathbf{a},\mathbf{v}) - \mathrm{Rev}(\mathbf{a}',\mathbf{v}) \leq n \max_{\ell \leq k} (a_\ell - a'_\ell).$$

To complete the proof, we argue by induction that $a_\ell - a'_\ell \leq \ell\epsilon$. This clearly holds for $\ell = 1$. For $\ell \geq 2$, the definition of a'_ℓ, in the absence of discretization to ϵ, would yield $a'_\ell = a_\ell - (a_{\ell-1} - a'_{\ell-1})$. With the discretization, we have

$$a'_\ell \geq a_\ell - (a_{\ell-1} - a'_{\ell-1}) - \epsilon \geq a_\ell - (\ell-1)\epsilon - \epsilon = a_\ell - \ell\epsilon,$$

where we used the inductive hypothesis at $\ell - 1$.

Next, we construct $\mathbf{a}''$ by setting $a''_\ell = \epsilon$ when $a'_\ell = 0$, and $a''_\ell = a'_\ell$ otherwise. We show that any bidder i that purchased some item ℓ in auction $\mathbf{a}'$ with price $a'_\ell \geq \epsilon$, purchases the item ℓ in auction $\mathbf{a}''$ as well.

Assume to the contrary that bidder i purchases another item ℓ'. Since there is infinite supply, bidder i could have purchased item ℓ' in auction $\mathbf{a}'$, but preferred to purchase item ℓ. Therefore,

$$v_i(\mathbf{e}_\ell) - a'_\ell \geq v_i(\mathbf{e}_{\ell'}) - a'_{\ell'}.$$

Note that $a''_\ell = a'_\ell$ and $a''_{\ell'} \geq a'_{\ell'}$ for all $\ell' \neq \ell$. So,

$$v_i(\mathbf{e}_\ell) - a''_\ell \geq v_i(\mathbf{e}_{\ell'}) - a''_{\ell'}.$$

Therefore, bidder i purchases item ℓ in auction $\mathbf{a}''$ as well.

Since only the bidders who receive items at price 0 might not be allocated the same items, we have that $\text{Rev}(\mathbf{a}'', \mathbf{v}) \geq \text{Rev}(\mathbf{a}', \mathbf{v})$. This completes the proof. □

C.4 Proof of Lemma 3.12

Since $\Gamma^{\text{SL}}_{\theta,\mathbf{v}} = \text{Rev}(\boldsymbol{\theta}, \mathbf{v})$, $\boldsymbol{\Gamma}^{\text{SL}}$ can be implemented by datasets $S_{\mathbf{v}} = \{(1, \mathbf{v})\}$ for $\mathbf{v} \in V$. So, $\boldsymbol{\Gamma}$ is implementable with complexity 1.

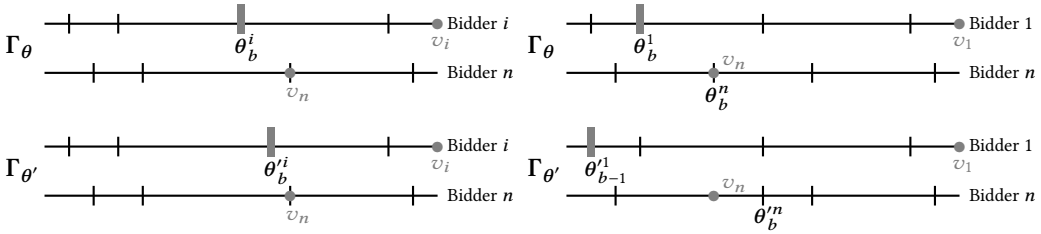

Fig. 3. Examples of cases 1 and 2 in the proof of Lemma 3.12. Bidder valuations are depicted as gray circles on the real line and the revenues of the two auctions $\boldsymbol{\theta}$ and $\boldsymbol{\theta}'$ are depicted as gray solid vertical lines.

It remains to show admissibility. Take any two different auctions $\boldsymbol{\theta}$ and $\boldsymbol{\theta}'$. We will show that $\boldsymbol{\Gamma}^{\text{SL}}_{\boldsymbol{\theta}} \neq \boldsymbol{\Gamma}^{\text{SL}}_{\boldsymbol{\theta}'}$. Let b be the smallest level at which there is $i \in [n]$ such that $\theta^i_b \neq \theta'^i_b$ and among such i choose the largest. There are three cases (see Figure 3 for cases 1 and 2):

(1) $i < n$: Consider the bid profile $\mathbf{v}^{i,\ell}$ for $\ell = m\theta^n_b$, in which bidder n's bid equals $\theta^n_b = \theta'^n_b$, meaning that his level in both $\boldsymbol{\theta}$ and $\boldsymbol{\theta}'$ is $b \geq 0$. Since bidder i's bid equals 1, his level is $s - 1 \geq 1$ under any auction, and bidders other than n and i have levels equal to 0 or -1. Thus, bidder i wins the item in both $\boldsymbol{\theta}$ and $\boldsymbol{\theta}'$, and pays the b^{th} threshold. So, $\text{Rev}(\boldsymbol{\theta}, \mathbf{v}^{i,\ell}) = \theta^i_b \neq \theta'^i_b = \text{Rev}(\boldsymbol{\theta}', \mathbf{v}^{i,\ell})$.

(2) $i = n$ and $b \geq 1$: Without loss of generality, assume that $\theta^n_b < \theta'^n_b$. Let $\ell = m\theta^n_b$ and consider $\mathbf{v}^{1,\ell}$. Bidder n's bid equals θ^n_b, so his level under $\boldsymbol{\theta}$ is b, whereas his level under $\boldsymbol{\theta}'$ is $b - 1 \geq 0$ (since all thresholds below b are the same in both auctions). So, bidder 1 wins the item in both auctions and pays the threshold that corresponds to bidder n's level. Therefore, $\text{Rev}(\boldsymbol{\theta}, \mathbf{v}^{1,\ell}) = \theta^1_b \neq \theta^1_{b-1} = \theta'^1_{b-1} = \text{Rev}(\boldsymbol{\theta}', \mathbf{v}^{1,\ell})$.

(3) $i = n$ and $b = 0$: Consider bid profile $\mathbf{e}_n$. In this profile, bidder n wins and pays the reserve price, which is equal to his threshold at $b = 0$. Therefore, $\text{Rev}(\boldsymbol{\theta}, \mathbf{e}_n) = \theta^n_0 \neq \theta'^n_0 = \text{Rev}(\boldsymbol{\theta}', \mathbf{e}_n)$.

Therefore, $\boldsymbol{\Gamma}^{\text{SL}}_{\boldsymbol{\theta}} \neq \boldsymbol{\Gamma}^{\text{SL}}_{\boldsymbol{\theta}'}$. Since any element of $\boldsymbol{\Gamma}^{\text{SL}}$ is a multiple of $1/m$, $\boldsymbol{\Gamma}^{\text{SL}}$ is $1/m$-admissible. □

D OMITTED PROOFS FROM SECTION 4

D.1 Proof of Lemma 4.1

Let $x^* = \arg\max_{x \in \mathcal{X}} \mathbb{E}_{y \sim F}[f(x, y)]$. By the definition of regret we have that for any $y_1, \ldots, y_T$,

$$\sum_{t=1}^{T} \mathbb{E}_{x_t}[f(x_t, y_t)] = \sup_{x \in \mathcal{X}} \sum_{t=1}^{T} f(x, y_t) - \textsc{Regret} \geq \sum_{t=1}^{T} f(x^*, y_t) - \textsc{Regret}.$$

Observe that the random variables $Z_t = f(x^*, y_t)$ are drawn i.i.d. with expected value $\mathbb{E}_{y \sim F}[f(x^*, y)]$ and are bounded in $[0, 1]$. Hence, by the Hoeffding bound, we get that with probability at least $1 - e^{-2T\epsilon^2}$:

$$\frac{1}{T} \sum_{t=1}^{T} f(x^*, y_t) \geq \mathbb{E}_{y \sim F}[f(x^*, y)] - \epsilon. \tag{10}$$

By setting $\epsilon = \sqrt{\frac{\log(1/\xi)}{2T}}$ and combining the two bounds we get the result. □

D.2 Proof of Lemma 4.3

Let $x^* = \arg\max_{x\in\mathcal{X}} \mathbb{E}_{y\sim F}\left[f(x,y)\right]$. As in the proof Lemma 4.1,

$$\sum_{t=1}^{T} \mathbb{E}_{x_t}[f(x_t,y_t)] = \sup_{x\in\mathcal{X}} \sum_{t=1}^{T} f(x,y_t) - \textsc{Regret} \geq \sum_{t=1}^{T} f(x^*,y_t) - \textsc{Regret}.$$

Since $y_1,\ldots,y_T$ are a Markov chain, by applying Theorem 4.2 to this chain and to the function $f(x^*,\cdot)$, we obtain that with probability at least $1-2\exp\left(-\frac{T\gamma\epsilon^2}{4+10\epsilon}\right)$,

$$\frac{1}{T}\sum_{t=1}^{T} f(x^*,y_t) \geq \mathbb{E}_{y\sim F}[f(x^*,y)] - \epsilon. \tag{11}$$

If we set $\epsilon = \sqrt{\frac{14\log(2/\xi)}{\gamma T}}$ then we have, either $\epsilon > 1$, in which case the inequality is trivial, since $f(x,y) \in [0,1]$, or if $\epsilon \leq 1$, then $\epsilon = \sqrt{\frac{14\log(2/\xi)}{\gamma T}} \geq \sqrt{\frac{(4+10\epsilon)\log(2/\xi)}{\gamma T}}$, which implies that the inequality holds with probability $1-\xi$. □

E OMITTED PROOFS FROM SECTION 5

E.1 Proof of Lemma 5.4

First we argue that $\mathbf{\Gamma}^Q$ has distinct rows. We will show that for any two rows π, π', there exists a column (σ,j) on which they differ. Since Q is a separator, there exists a context $\sigma^* \in Q$ on which $\pi(\sigma^*) \neq \pi'(\sigma^*)$. Now by the admissibility of the original matrix $\mathbf{\Gamma}$, we know that for any two $x,x' \in \mathcal{X}$, there exists a column j of the original matrix such that $\Gamma_{x,j} \neq \Gamma_{x',j}$. Applying the latter for $\pi(\sigma^*),\pi'(\sigma^*) \in \mathcal{X}$, we get that there exists a j^*, such that $\Gamma_{\pi(\sigma^*),j^*} \neq \Gamma_{\pi'(\sigma^*),j^*}$. By the definition of $\mathbf{\Gamma}^Q$, the latter implies that $\Gamma^Q_{\pi,(\sigma^*,j^*)} \neq \Gamma^Q_{\pi',(\sigma^*,j^*)}$. Thus the two rows π,π' of matrix $\mathbf{\Gamma}^Q$ differ at column (σ^*,j^*).

We now argue that if the quantity δ from Definition 2.2 is valid for $\mathbf{\Gamma}$, it is also valid for $\mathbf{\Gamma}^Q$. We remind the reader that δ is required to lower bound the minimum positive difference between any two elements of a column. Since the values appearing in the column (σ,j) of matrix $\mathbf{\Gamma}^Q$ are all taken from the column j of matrix $\mathbf{\Gamma}$, any lower bound δ that is valid for $\mathbf{\Gamma}$ is also valid for $\mathbf{\Gamma}^Q$. Thus $\mathbf{\Gamma}^Q$ is also δ-admissible. □

E.2 Proof of Lemma 5.5

The intuition of the proof is as follows: if we know that we can simulate every column in $\mathbf{\Gamma}$ with a sequence of polynomially many weighted inputs $\{(w,y)\}$, then we can simulate each column of $\mathbf{\Gamma}^Q$ associated with context $\sigma \in Q$ and column j of $\mathbf{\Gamma}$, by a sequence of weighted contextual inputs $\{(w,(\sigma,y))\}$, which is essentially a contextually annotated copy of the sequence of inputs we used to simulate column j of $\mathbf{\Gamma}$. We now show this intuition more formally.

Since $\mathbf{\Gamma}$ is implementable with complexity M, we have that for any column $j \in \{1,\ldots,N\}$ of matrix $\mathbf{\Gamma}$ there exists a weighted dataset $S_j = \{(w,y)\}$, with $|S_j| \leq M$, such that:

$$\text{for all } x,x' \in \mathcal{X}: \quad \Gamma_{xj} - \Gamma_{x'j} = \sum_{(w,y)\in S_j} w\cdot\Big(f(x,y) - f(x',y)\Big).$$

For the contextual problem, we need to argue that for any column $(\sigma, j) \in Q \times [N]$ of matrix $\mathbf{\Gamma}^Q$ there exists a weighted contextual dataset $S^c_{\sigma,j} = \{ (w, (\sigma, y)) \}$, with $|S^c_{\sigma,j}| \leq M$, such that:

$$\text{for all } \pi, \pi' \in \Pi: \quad \Gamma^Q_{\pi,(\sigma,j)} - \Gamma^Q_{\pi',(\sigma,j)} = \sum_{(w,(\sigma,y)) \in S^c_{\sigma,j}} w \cdot \Big(f(\pi(\sigma), y) - f(\pi'(\sigma), y) \Big).$$

The construction of such a contextual dataset is straightforward: for each $(w, y) \in S_j$, create a data point $\Big(w, (\sigma, y)\Big)$ in $S^c_{\sigma,j}$. To finish the proof, observe that

$$\begin{aligned} \Gamma^Q_{\pi,(\sigma,j)} - \Gamma^Q_{\pi',(\sigma,j)} = \Gamma_{\pi(\sigma),j} - \Gamma_{\pi'(\sigma),j} &= \sum_{(w,y) \in S_j} w \cdot \Big(f(\pi(\sigma), y) - f(\pi'(\sigma), y) \Big) \\ &= \sum_{(w,(\sigma,y)) \in S^c_{\sigma,j}} w \cdot \Big(f(\pi(\sigma), y) - f(\pi'(\sigma), y) \Big). \end{aligned}$$

□

E.3 Proof of Lemma 5.8

We will show that for each $t \leq T$,

$$\mathbb{E}[f_c(\pi_{t+1}, (\sigma_t, y_t)) - f_c(\pi_t, (\sigma_t, y_t))] \leq 2N\rho(1 + \delta^{-1}). \tag{12}$$

Since $f_c(\pi, (\sigma_t, y_t)) = f(\pi(\sigma_t), y_t)$ and since $f(x, y) \in [0, 1]$, it suffices to show that $\Pr[\pi_{t+1}(\sigma_t) \neq \pi_t(\sigma_t)] \leq 2N\rho(1 + \delta^{-1})$.

Observe that if $\pi_{t+1}(\sigma_t) \neq \pi_t(\sigma_t)$ then it must be that $\Gamma_{\pi_{t+1}(\sigma_t),j} \neq \Gamma_{\pi_t(\sigma_t),j}$ for some $j \in [N]$, by the admissibility of matrix $\mathbf{\Gamma}$ for the non-contextual problem. Thus we need to show that the probability that there exists a j such that $\Gamma_{\pi_{t+1}(\sigma_t),j} \neq \Gamma_{\pi_t(\sigma_t),j}$ is at most $2N\rho(1 + \delta^{-1})$. By the union bound it suffices to show that for any given j, the probability that $\Gamma_{\pi_{t+1}(\sigma_t),j} \neq \Gamma_{\pi_t(\sigma_t),j}$ is at most $2\rho(1 + \delta^{-1})$.

The proof of this fact then follows by identical arguments as in the proof of the non-contextual stability in Lemma 2.4 and we omit it for succinctness. □

E.4 Proof of Theorem 5.9

Using Lemma 5.8, and since D is uniform over $[-v, v]$ and is $\left(\frac{1+2\epsilon}{2v\delta}, \frac{1+2\epsilon}{\delta}\right)$-dispersed, we have:

$$\mathbb{E}\left[\sum_{t=1}^{T} f(x_t, y_t) - f(x_{t+1}, y_t)\right] \leq \frac{TN(1 + 2\epsilon)(1 + \delta)}{v\delta^2} \quad .$$

It remains to bound the term $\mathbb{E}\left[\sum_{\sigma \in P} \sum_{j=1}^{N} \alpha_{\sigma,j} (\Gamma^P_{\pi^*,(\sigma,j)} - \Gamma^P_{\pi_1,(\sigma,j)})\right]$, which is at most

$$2\mathbb{E}\left[\max_{\pi} \sum_{\sigma \in P} \sum_{j=1}^{N} \alpha_{\sigma,j} \Gamma^P_{\pi,(\sigma,j)}\right].$$

Let $\beta_\pi = \sum_{\sigma \in P} \sum_{j=1}^{N} \alpha_{\sigma,j} \Gamma^P_{\pi,(\sigma,j)}$. We therefore have for any $\lambda > 0$,

$$\begin{aligned} \mathbb{E}\left[\max_{\pi} \beta_\pi\right] &= \tfrac{1}{\lambda} \ln\left(\exp\left(\lambda \mathbb{E}\left[\max_{\pi} \beta_\pi\right]\right)\right) \\ &\leq \tfrac{1}{\lambda} \ln\left(\mathbb{E}\left[\exp\left(\lambda \max_{\pi} \beta_\pi\right)\right]\right) && \text{(Jensen's inequality)} \\ &\leq \tfrac{1}{\lambda} \ln\left(\sum_{\pi} \mathbb{E}\left[\exp\left(\lambda \beta_\pi\right)\right]\right) \end{aligned}$$

$$
\begin{aligned}
&= \tfrac{1}{\lambda}\ln\left(\sum_{\pi}\prod_{\sigma,j}\mathbb{E}\left[\exp\left(\lambda\alpha_{\sigma,j}\Gamma^{P}_{\pi,(\sigma,j)}\right)\right]\right) && \text{(by independence of } \alpha_{\sigma,j}\text{)}\\
&\le \tfrac{1}{\lambda}\ln\left(\sum_{\pi}\prod_{\sigma,j}\exp\left(\tfrac{\nu^2\lambda^2}{2}\right)\right) && \text{(Hoeffding's lemma for bounded r.v.)}\\
&= \frac{\ln|\Pi|}{\lambda}+\frac{\nu^2 N|P|\lambda}{2}\\
&= \nu\sqrt{2N|P|\ln|\Pi|}\ . && \text{(by picking the optimal } \lambda\text{)}
\end{aligned}
$$

The theorem now follows by combining the stability and the error bound above and invoking Lemma 2.1. □

E.5 Proof of Corollary 5.13

According to Lemma 2.1, it suffices to bound the stability term $\mathbb{E}\left[\sum_{t=1}^{T} f_c(\pi_{t+1},(\sigma_t,y_t))-f_c(\pi_t,(\sigma_t,y_t))\right]$ and the perturbation error term $\mathbb{E}\left[\boldsymbol{\alpha}\cdot(\boldsymbol{\Gamma}^{Q}_{\pi_1}-\boldsymbol{\Gamma}^{Q}_{\pi^*})\right]$. Since the distribution D is $\left(\frac{1+2\epsilon}{2\nu\delta},\frac{1+2\epsilon}{\delta}\right)$-dispersed and also $\boldsymbol{\Gamma}^{Q}$ is δ-admissible according to Lemma 5.4, we invoke Lemma 2.4 and plug in $\epsilon = 1/\sqrt{T}$ to conclude that the stability term is bounded by $\frac{TNd(1+2T^{-1/2})(1+\delta^{-1})}{\nu\delta}$. On the other hand, by the exact same proof as in Theorem 5.9, the perturbation error term is bounded by $2\nu\sqrt{2Nd\ln|\Pi|}$. Finally plugging in the value of ν (which optimizes the regret bound) finishes the proof. □

F PSEUDO-POLYNOMIAL AND INTEGER-WEIGHTED ORACLES

F.1 Pseudo-Polynomial Oracles

If the offline optpmization oracle is only a pseudo-polynomial-time approximation scheme, such as some combinatorial optimization methods or optimization methods based on gradient descent, then its running time will depend not only on the cardinality of the weighted data set S, but also on the actual magnitude of the weights. Therefore, the magnitude of the weights that are needed to implement matrix $\boldsymbol{\Gamma}$ will influence the final running time of the learning algorithm. To capture such types of oracles, we introduce the notion of the *pseudo-complexity*:

Definition F.1. *The* pseudo-complexity *of a weighted data set* $S=\{(w_\ell,y_\ell)\}_{\ell\in\mathcal{L}}$ *with* $w_\ell\in\mathbb{R}^+$ *is*

$$\text{pseudo}(S) := \max\left\{|S|,\ \textstyle\sum_{\ell\in\mathcal{L}} w_\ell\right\}.$$

Definition F.2. *A matrix* $\boldsymbol{\Gamma}$ *is implementable with* pseudo-complexity W *if for each* $j\in[N]$ *there exists a weighted dataset* S_j*, with* $\text{pseudo}(S_j)\le W$*, such that*

$$\text{for all } x,x'\in\mathcal{X}\text{:}\quad \Gamma_{xj}-\Gamma_{x'j}=\sum_{(w,y)\in S_j} w\Big(f(x,y)-f(x',y)\Big).$$

In this case, we say that weighted datasets S_j, $j\in[N]$, *implement* $\boldsymbol{\Gamma}$ *with pseudo-complexity* W.

As an immediate corollary, we obtain that the existence of a pseudo-polynomial-time offline oracle implies the existence of a polynomial-time online learner with regret $O(\sqrt{T})$, whenever we have access to an implementable and admissible matrix. The following is an immediate consequence of Theorem 2.5 used with pseudo-polynomial oracles.

Corollary F.3. *Consider the problem $(\mathcal{X},\mathcal{Y},f)$ where $f:\mathcal{X}\times\mathcal{Y}\to[0,1]$. Assume that $\mathbf{\Gamma}\in[0,1]^{|\mathcal{X}|\times N}$ is implementable with pseudo-complexity W and δ-admissible, and there exists an approximate offline oracle $\textsc{Opt}\left(S,\frac{1}{\sqrt{T}}\right)$ which runs in time $\mathrm{poly}(\mathrm{pseudo}(S),T)$. Then Algorithm 2 with distribution D as defined in Theorem 2.5 runs in time $\mathrm{poly}(N,W,T,1/\delta)$ and achieves regret $O(N\sqrt{T}/\delta)$.*

Note that for some problem classes $(\mathcal{X},\mathcal{Y},f)$ discussed in the main body of the paper, such as multi-item or multi-unit auctions, we have $f:\mathcal{X}\times\mathcal{Y}\to[0,R]$ for $R>1$. As we briefly discussed in Section 3, to apply our results we first implicitly re-scale function f to f' such that $f'(x,y)=f(x,y)/R$ for all $x\in\mathcal{X}$ and $y\in\mathcal{Y}$. We then apply Theorem 2.5 to the problem $(\mathcal{X},\mathcal{Y},f')$ and obtain a $O(N\sqrt{T}/\delta)$ regret for the normalized problem. Lastly, we scale up f' back to f and get a regret bound that is R times the regret for the normalized problem, i.e., $O(RN\sqrt{T}/\delta)$ for the problem $(\mathcal{X},\mathcal{Y},f)$. Let $\mathbf{\Gamma}$ be the δ-admissible matrix for the original problem $(\mathcal{X},\mathcal{Y},f)$ that is implemented by $S=\{(w_\ell,y_\ell)\}_{\ell\in\mathcal{L}}$. Then, $S'=\{(w_\ell R,y_\ell)\}_{\ell\in\mathcal{L}}$ implements $\mathbf{\Gamma}$ for the re-scaled problem $(\mathcal{X},\mathcal{Y},f')$. That is, the pseudo-complexity of implementing $\mathbf{\Gamma}$ through this re-scaling process is increased by at most a factor of R. Since R is polynomial in the parameters of interest in all of our applications, this does not affect the running time of our algorithms substantially. Formally, we rely on the following result.

Corollary F.4. *Consider the problem $(\mathcal{X},\mathcal{Y},f)$ where $f:\mathcal{X}\times\mathcal{Y}\to[0,R]$, $R\geq 1$. Assume that $\mathbf{\Gamma}\in[0,1]^{|\mathcal{X}|\times N}$ is implementable with pseudo-complexity W for $(\mathcal{X},\mathcal{Y},f)$ and δ-admissible, and there exists an approximate offline oracle $\textsc{Opt}\left(S,\frac{1}{\sqrt{T}}\right)$ which runs in time $\mathrm{poly}(\mathrm{pseudo}(S),T)$. Then there is an algorithm that runs in time $\mathrm{poly}(N,R,W,T,1/\delta)$ and achieves regret $O(RN\sqrt{T}/\delta)$.*

In Appendix F.4, we derive the pseudo-complexity of implementing matrices $\mathbf{\Gamma}$ for the (unnormalized) applications discussed in this paper. In all cases, the pseudo-complexity (W) of the unnormalized problem multiplied by its range (R) either agrees with complexity (M) of the problem or is a small function of the complexity and the number of columns. Thus, in all these applications we obtain online algorithms with runtime $\mathrm{poly}(N,M,T,1/\delta)$ with only requiring pseudo-polynomial offline algorithms. Furthermore, in all auction applications discussed in this paper $1/\delta=\mathrm{poly}(N,M)$ leading to online algorithms with runtime $\mathrm{poly}(N,M,T)$ that only require pseudo-polynomial offline algorithms.

F.2 Integer-Weighted Oracles

In the main body of the paper, we consider oracles Opt that take as input a *real-weighted* dataset and a precision parameter ϵ. In this section, we focus on a seemingly more restricted oracle, IntOpt, that only takes integer-weighted datasets. We show that all of our results in Section 2 extend to this class of oracles.

Definition F.5 (Integer-Weighted Offline Oracle). *An* integer-weighted offline oracle IntOpt *receives as input a set of adversary actions with non-negative integer weights $S=\{(w_\ell,y_\ell)\}_{\ell\in\mathcal{L}}$ with $w_\ell\in\mathbb{N}$, $y_\ell\in\mathcal{Y}$ and a desired precision ϵ, and returns an action $\hat{x}=\textsc{IntOpt}(S,\epsilon)$ such that*

$$\sum_{(w,y)\in S} wf(\hat{x},y)\geq\max_{x\in\mathcal{X}}\sum_{(w,y)\in S} wf(x,y)-\epsilon.$$

We show that IntOpt can be used to implement a real-weighted oracle Opt. This implies that online learning can be efficiently reduced to offline optimization with IntOpt oracles under the same conditions as studied in Section 2. Next lemma outlines the construction of Opt from IntOpt. The resulting variant of the the Oracle-based Generalized FTPL algorithm (Algorithm 2) is provided in Algorithm 4. Note that the constructed Opt inherits the polynomial or the pseudo-polynomial

Algorithm 4: Oracle-Based Generalized FTPL with Integer-Weighted Oracle

Input: datasets S_j, $j \in [N]$, that implement a matrix $\mathbf{\Gamma} \in [0,1]^{|\mathcal{X}| \times N}$,
distribution D over $\mathbb{R}^+$,
an integer-weighted offline oracle IntOpt.
Draw $\alpha_j \sim D$ independently for $j = 1, \ldots, N$.
For all j, let $\alpha_j S_j$ denote the scaled version of S_j, i.e., $\alpha_j S_j \coloneqq \{(\alpha_j w, y) : (w,y) \in S_j\}$.
for $t = 1, \ldots, T$ **do**
Set $S = \left\{(1, y_1), \ldots, (1, y_{t-1})\right\} \cup \bigcup_{j \le N} \alpha_j S_j$.
Set $S' = \left\{(w', y) : \; w' = \lfloor w \cdot 2|S| \sqrt{T} \rfloor \text{ and } (w,y) \in S\right\}$.
Play $x_t = \textsc{IntOpt}(S', |S'|)$.
Observe y_t and receive payoff $f(x_t, y_t)$.
end for

efficiency of IntOpt. Pseudo-polynomial running times may arise naturally for those integer-weighted oracles, in which the integer weights are implemented simply by replicated examples.

Lemma F.6 (Integer-Weighted Offline Oracle to Offline Oracle). *Given any real-weighted dataset S, any precision parameter $\epsilon > 0$, and an integer-weighted offline oracle* IntOpt, *let*

$$S' = \{(w', y) : \; w' = \lfloor w \cdot 2|S|/\epsilon \rfloor \text{ and } (w,y) \in S\}, \quad \epsilon' = |S'|.$$

For all $x \in \mathcal{X}$, we have

$$\sum_{(w,y) \in S} w f(\textsc{IntOpt}(S', \epsilon'), y) \ge \sum_{(w,y) \in S} w f(x,y) - \epsilon.$$

Furthermore, $|S'| = |S|$ and $\text{pseudo}(S') \le 2\left(\text{pseudo}(S)\right)^2/\epsilon$. Therefore, if $\textsc{IntOpt}\left(S', |S'|\right)$ runs in time $\text{poly}(|S'|)$ or $\text{poly}(\text{pseudo}(S'))$, then the above construction with $\epsilon = 1/\sqrt{T}$ implements $\textsc{Opt}\left(S, \frac{1}{\sqrt{T}}\right)$ that runs in time $\text{poly}(|S|, T)$ or $\text{poly}(\text{pseudo}(S), T)$, respectively.

Proof. Let $\delta = \epsilon/(2|S|)$. Thus, elements $(w,y) \in S$ are transformed into $(\lfloor w/\delta \rfloor, y) \in S'$. To show the approximate optimality, we use the fact that $0 \le a - \lfloor a \rfloor \le 1$ for any $a \ge 0$, so we have $0 \le w - \delta \lfloor w/\delta \rfloor \le \delta$ and thus for any $x \in \mathcal{X}$,

$$\begin{aligned}
\sum_{(w,y) \in S} w f(\textsc{IntOpt}(S', \epsilon'), y) &\ge \delta \sum_{(w,y) \in S} \lfloor w/\delta \rfloor f(\textsc{IntOpt}(S', \epsilon'), y) \\
&= \delta \sum_{(w',y) \in S'} w' f(\textsc{IntOpt}(S', \epsilon'), y) && \text{(by construction)} \\
&\ge \delta \sum_{(w',y) \in S'} w' f(x,y) - \delta\epsilon' && \text{(from the definition of IntOpt)} \\
&\ge \sum_{(w,y) \in S} (w - \delta) f(x,y) - \delta\epsilon' \\
&\ge \sum_{(w,y) \in S} w f(x,y) - |S|\delta - |S|\delta \\
&= \sum_{(w,y) \in S} w f(x,y) - \epsilon. && \square
\end{aligned}$$

As an immediate consequence of Lemma F.6 and Corollaries 2.11 and F.4 we obtain the following:

THEOREM F.7. *Assume that $\mathbf{\Gamma} \in [0,1]^{|\mathcal{X}|\times N}$ is δ-admissible. Then Algorithm 4 with distribution D as defined in Theorem 2.5 achieves regret $O(N\sqrt{T}/\delta)$.*

Furthermore, if $\mathbf{\Gamma}$ is implementable with real-weighted datasets with complexity M and the integer-weighted oracle $\textsc{IntOpt}(S', |S'|)$ runs in time $\mathrm{poly}(|S'|)$ then Algorithm 4 runs in time $\mathrm{poly}(N, M, T)$. If $\mathbf{\Gamma}$ is implementable with real-weighted datasets with pseudo-complexity W and the integer-weighted oracle $\textsc{IntOpt}(S', |S'|)$ runs in time $\mathrm{poly}(\mathrm{pseudo}(S'))$ then Algorithm 4 runs in time $\mathrm{poly}(N, W, T, 1/\delta)$.

F.3 A Note on Numerical Computations

For the ease of exposition, we have assumed throughout the paper that all numerical computations take $O(1)$ time. However, to invoke a real-weighted oracle on a given weighted dataset requires to communicate the weights, which may, in principle, be of an unbounded description length. This problem can be circumvented by discretizing real weights to an appropriate accuracy as described in Appendix F.2 and instantiated in Algorithm 4. Specifically, during the run of Algorithm 4 with D as defined in Theorem 2.5 we generate integer-weighted datasets S with the pseudo-complexity $\mathrm{poly}(N, W, T, 1/\delta)$ for a δ-admissible matrix $\mathbf{\Gamma}$ with N columns implementable with pseudo-complexity W. Therefore, each individual weight can be described by $O(\log(NWT/\delta))$ bits. Similarly, when the function is in range $[0, R]$, as discussed in Section F.1, we generate integer-weighted datasets S with the pseudo-complexity $\mathrm{poly}(N, R, W, T, 1/\delta)$ for a δ-admissible matrix $\mathbf{\Gamma}$ with N columns implementable with pseudo-complexity W. In all of our applications, the pseudo-complexity of implementing $\mathbf{\Gamma}$, i.e., W, multiplied by the range of the utility function, i.e., R, is polynomial in the complexity M of implementing $\mathbf{\Gamma}$ and the number of columns N. Furthermore, the admissibility constant δ satisfies $1/\delta = \mathrm{poly}(N, M)$. As a result, each weight requires only $O(\log(NMT))$ bits, and the oracle-based running time of Algorithm 4 remains $\mathrm{poly}(N, M, T)$ regardless of whether we assume that the numerical computation and storage of reals are $O(1)$ or polynomial in the size of the bit representation.

F.4 Pseudo-Complexity of Applications

In this section, we provide the pseudo-complexity of implementing matrices that are used by the applications in Sections 3.1, 3.2, 3.3, 7.1 and 7.2. The results are summarized in Table 3. In our derivations below, recall that m denotes the discretization level used in these auction classes. To show that $\mathbf{\Gamma}$ is implementable with pseudo-complexity W, we need to show that the datasets S_j implementing its columns satisfy $\mathrm{pseudo}(S_j) \le W$, i.e., $|S_j| \le W$ and $\sum_{(w,y)\in S_j} w \le W$.

VCG with Bidder-Specific Reserves. We show that the pseudo-complexity of implementing $\mathbf{\Gamma}^{\mathrm{VCG}}$ is $O(m^2)$.

Recall that we used datasets S_j for each column $j \le n\lceil \log m \rceil$ that included the m bid profiles in which only the bidder corresponding to column j (call it bidder i) has non-zero valuation, denoted as $\mathbf{v}_h := (h/m)\mathbf{e}_i$ for $h \le m$. Specifically, we used $S_j = \{(w_h, \mathbf{v}_h)\}_{h\in[m]}$ with the weights defined recursively as follows:

$$w_m = \max\left\{0,\ \max_z \left[m\left(z_\beta - (z-1)_\beta\right)\right]\right\},$$

and for all $z = m, m-1, \ldots, 2$,

$$w_{z-1} = \frac{1}{z-1}\left(\sum_{h=z}^{m} w_h - m\left(z_\beta - (z-1)_\beta\right)\right).$$

Table 3. The pseudo-complexity and complexity of the datasets that implement translation matrices for our (un-normalized) applications. In this table, n refers to the number of bidders, m refers to the bid's discretization level, and k refers to the number of items. The payoff range $[0,R]$ for $R>1$ affects the true pseudo-complexity of querying the oracle by R. The payoff range of $[0,n]$ for VCG auction with bidder-specific reserves is stated for the general case where every bidder's valuation is in range $[0,1]$. For s-unit VCG auction with bidder-specific reserves the payoff range is $[0,s]$ for $s \le n$.

Problem class	Matrix	Pseudo-complexity	Complexity	Number of columns	Admissibility δ	Payoff range	Section
VCG with bidder-specific reserves	$\mathbf{\Gamma}^{\text{VCG}}$	$O(m^2)$	m	$n\lceil \log m \rceil$	1	$[0,n]$	3.1
envy-free item pricing	$\mathbf{\Gamma}^{\text{IP}}$	$O(m^2)$	m	$k\lceil \log m \rceil$	1	$[0,n]$	3.2
level auctions (without repetitions)	$\mathbf{\Gamma}^{\text{SL}}$	1	1	nm	$1/m$	$[0,1]$	3.3
level auctions (with repetitions)	$\mathbf{\Gamma}^{\overline{\text{RL}}}$	1	1	$O(n^3m^3)$	$1/m$	$[0,1]$	3.3
multi-unit welfare maximization	$\mathbf{\Gamma}^{\text{MU}}$	1	1	n	$1/(4n^2)$	$[0,n]$	7.1
SiSPAs	$\mathbf{\Gamma}^{\text{OB}}$	m	m	k	$1/m$	$[0,k]$	7.2

We show that $\sum_{z=1}^{m} w_z \le 2m(m-1)+m$. Note that by definition $w_m \le m$. Moreover, by the definition of w_{z-1}, we have

$$w_{z-1} = \frac{1}{z-1}\left(\sum_{h=z}^{m} w_h - m\Big(z_\beta - (z-1)_\beta\Big)\right) \le \frac{1}{z-1}\left(\sum_{h=z}^{m} w_h + m\right).$$

Reformulating the above, we have

$$(z-1)w_{z-1} \le \sum_{h=z}^{m} w_h + m \le \sum_{h=z}^{m-1} w_h + 2m.$$

Next, we sum over the above inequality for $z = 2,\dots,m$, obtaining

$$\begin{aligned}
\sum_{z=2}^{m}(z-1)w_{z-1} &\le \sum_{z=2}^{m}\left(\sum_{h=z}^{m-1} w_h + 2m\right) \\
&= \sum_{z=2}^{m}\sum_{h=z}^{m-1} w_h + 2m(m-1) \\
&= \sum_{h'=2}^{m-1}(h'-1)w_{h'} + 2m(m-1) \\
&= \sum_{h'=3}^{m}(h'-2)w_{h'-1} + 2m(m-1),
\end{aligned}$$

where the penultimate transition is by rearranging the summation and counting the number of times each $w_{h'}$ appears in the sum. Subtracting the two sides and changing parameter $z \leftarrow z-1$, we get

$$\sum_{z=1}^{m-1} w_z \le 2m(m-1).$$

Thus, using the fact that $w_m \le m$, we obtain $\sum_{z=1}^{m} w_z \le 2m(m-1)+m$. Since $|S_j| = m$, this means that $\text{pseudo}(S_j) = O(m^2)$.

Envy-free Item Pricing. As we show in Appendix C.2, as far as the implementability of $\mathbf{\Gamma}^{\text{IP}}$ is concerned, this setting is isomorphic to the case of VCG with bidder-specific prices. So, the pseudo-complexity of implementing $\mathbf{\Gamma}^{\text{IP}}$ is also $O(m^2)$.

Level Auctions. Recall that the datasets used for implementing $\mathbf{\Gamma}^{\mathrm{SL}}$ are of the form of $S_{\mathbf{v}} = \{(1, \mathbf{v})\}$, for a suitable $\mathbf{v}$, so $\mathbf{\Gamma}^{\mathrm{SL}}$ is implementable with pseudo-complexity of 1.

Similarly, recall that $\mathbf{\Gamma}^{\overline{\mathrm{RL}}}$ is a matrix whose columns are indexed by $\mathbf{v} \in \{\mathbf{v} \mid \mathbf{v} \in \{0, 1/m, \ldots, m/m\}^n$ and $\|\mathbf{v}\|_0 \leq 3\}$, such that $\Gamma^{\overline{\mathrm{RL}}}_{\boldsymbol{\theta},\mathbf{v}} = \mathrm{Rev}(\boldsymbol{\theta}, \mathbf{v})$. That is, $\mathbf{\Gamma}^{\overline{\mathrm{RL}}}$ is implemented by datasets $S_{\mathbf{v}} = \{(1, \mathbf{v})\}$. Therefore $\mathbf{\Gamma}^{\overline{\mathrm{RL}}}$ is implementable with pseudo-complexity of 1.

Multi-unit Welfare Maximization. Recall that $\mathbf{\Gamma}^{\mathrm{MU}}$ is implemented by sets $S_j = \{(1, \mathbf{v})\}$, for a suitable $\mathbf{v}$, so $\mathbf{\Gamma}^{\mathrm{MU}}$ is implementable with pseudo-complexity of 1.

Online Bidding in SiSPAs. Recall that $\mathbf{\Gamma}^{\mathrm{OB}}$ is implemented by datasets $S_j = \{(w_\ell, \mathbf{p}_\ell)\}_{\ell \in \{0,\ldots,m-1\}}$, where

$$w_\ell = \begin{cases} \frac{1}{m} \cdot \frac{1}{v(\mathbf{e}_j) - \ell/m} & \text{if } \ell/m < v(\mathbf{e}_j) \\ 0 & \text{otherwise.} \end{cases}$$

Since $v(\mathbf{e}_j) \in \{0, 1/m, \ldots, m/m\}$, we have that $w_\ell \leq 1$. Moreover, $|S_j| = m$, so $\mathbf{\Gamma}$ is implementable with pseudo-complexity m.

ACKNOWLEDGMENTS

Part of this work was done when Haghtalab and Luo worked at Microsoft Research, New York. This work was partially supported by NSF under award IIS-1755781, a J.P. Morgan and Chase faculty fellowship, a Siebel scholarship, and a Microsoft Research Ph.D. fellowship.

REFERENCES

[1] Alekh Agarwal, Daniel Hsu, Satyen Kale, John Langford, Lihong Li, and Robert Schapire. 2014. Taming the monster: A fast and simple algorithm for contextual bandits. In *Proceedings of the 31st International Conference on Machine Learning (ICML)*. 1638–1646.

[2] Baruch Awerbuch and Robert Kleinberg. 2008. Online Linear Optimization and Adaptive Routing. *J. Comput. System Sci.* 74, 1 (Feb. 2008), 97–114.

[3] Maria-Florina Balcan and Avrim Blum. 2006. Approximation algorithms and online mechanisms for item pricing. In *Proceedings of the 7th ACM Conference on Economics and Computation (EC)*. ACM, 29–35.

[4] Kshipra Bhawalkar and Tim Roughgarden. 2011. Welfare guarantees for combinatorial auctions with item bidding. In *Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. SIAM, 700–709.

[5] Avrim Blum and Jason D Hartline. 2005. Near-optimal online auctions. In *Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. SIAM, 1156–1163.

[6] Avrim Blum and Yishay Mansour. 2007. Learning, Regret Minimization, and Equilibria. In *Algorithmic Game Theory*, Noam Nisan, Tim Roughgarden, Eva Tardos, and Vijay Vazirani (Eds.). Cambridge University Press, Chapter 4.

[7] Sébastien Bubeck, Nikhil R. Devanur, Zhiyi Huang, and Rad Niazadeh. 2019. Multi-scale Online Learning: Theory and Applications to Online Auctions and Pricing. *Journal of Machine Learning Research (JMLR)* 20, 62 (2019), 1–37.

[8] Nicolo Cesa-Bianchi, Alex Conconi, and Claudio Gentile. 2004. On the generalization ability of on-line learning algorithms. *IEEE Transactions on Information Theory* 50, 9 (2004), 2050–2057.

[9] Nicolo Cesa-Bianchi, Claudio Gentile, and Yishay Mansour. 2013. Regret minimization for reserve prices in second-price auctions. In *Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. Society for Industrial and Applied Mathematics, 1190–1204.

[10] Nicolo Cesa-Bianchi and Ohad Shamir. 2011. Efficient online learning via randomized rounding. In *Proceedings of the 25th Annual Conference on Neural Information Processing Systems (NeurIPS)*. 343–351.

[11] Jeff Cheeger. 1969. *A lower bound for the smallest eigenvalue of the Laplacian.* 195–199.

[12] George Christodoulou, Annamária Kovács, and Michael Schapira. 2008. Bayesian combinatorial auctions. In *Proceedings of the 35th International Colloquium on Automata, Languages and Programming (ICALP)*. Springer, 820–832.

[13] Richard Cole and Tim Roughgarden. 2014. The Sample Complexity of Revenue Maximization. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing (STOC)*. ACM, 243–252.

[14] Constantinos Daskalakis and Vasilis Syrgkanis. 2016. Learning in auctions: Regret is hard, envy is easy. In *Proceedings of the 57th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 219–228.

[15] Mahsa Derakhshan, Negin Golrezaei, and Renato Paes Leme. 2019. LP-Based Approximation for Personalized Reserve Prices. In *Proceedings of the 20th ACM Conference on Economics and Computation (EC)*. Association for Computing Machinery, 589.

[16] Nikhil R. Devanur, Zhiyi Huang, and Christos-Alexandros Psomas. 2017. The Sample Complexity of Auctions with Side Information. *CoRR* abs/1511.02296v4 (2017).

[17] Nikhil R. Devanur, Zhiyi Huang, and Christos-Alexandros Psomas. 2016. The Sample Complexity of Auctions with Side Information. In *Proceedings of the 48th Annual ACM Symposium on Theory of Computing (STOC)*. ACM, 426–439.

[18] Peerapong Dhangwatnotai, Tim Roughgarden, and Qiqi Yan. 2015. Revenue maximization with a single sample. *Games and Economic Behavior* 91 (2015), 318–333.

[19] Shahar Dobzinski and Noam Nisan. 2007. Mechanisms for multi-unit auctions. In *Proceedings of the 8th ACM Conference on Economics and Computation (EC)*. ACM, 346–351.

[20] Miroslav Dudík, Nika Haghtalab, Haipeng Luo, Robert E Schapire, Vasilis Syrgkanis, and Jennifer Wortman Vaughan. 2017. Oracle-Efficient Online Learning and Auction Design. In *Proceedings of the 58th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 528–539.

[21] Miroslav Dudik, Daniel Hsu, Satyen Kale, Nikos Karampatziakis, John Langford, Lev Reyzin, and Tong Zhang. 2011. Efficient optimal learning for contextual bandits. In *Proceedings of theAnnual Conference on Uncertainty in Artificial Intelligence (UAI)*. 169–178.

[22] Edith Elkind. 2007. Designing and learning optimal finite support auctions. In *Proceedings of the 8th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. Society for Industrial and Applied Mathematics, 736–745.

[23] Michal Feldman, Hu Fu, Nick Gravin, and Brendan Lucier. 2013. Simultaneous auctions are (almost) efficient. In *Proceedings of the 45th Annual ACM Symposium on Theory of Computing (STOC)*. ACM, 201–210.

[24] Yoav Freund and Robert E Schapire. 1997. A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. System Sci.* 55, 1 (1997), 119–139.

[25] Sally A Goldman, Michael J Kearns, and Robert E Schapire. 1993. Exact identification of read-once formulas using fixed points of amplification functions. *SIAM J. Comput.* 22, 4 (1993), 705–726.

[26] Venkatesan Guruswami, Jason D Hartline, Anna R Karlin, David Kempe, Claire Kenyon, and Frank McSherry. 2005. On profit-maximizing envy-free pricing. In *Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. SIAM, 1164–1173.

[27] James Hannan. 1957. Approximation to Bayes risk in repeated play. *Contributions to the Theory of Games* 3 (1957), 97–139.

[28] Jason D Hartline and Tim Roughgarden. 2009. Simple versus optimal mechanisms. In *Proceedings of the 10th ACM Conference on Economics and Computation (EC)*. ACM, 225–234.

[29] Elad Hazan and Satyen Kale. 2012. Online submodular minimization. *Journal of Machine Learning Research (JMLR)* 13, Oct (2012), 2903–2922.

[30] Elad Hazan and Tomer Koren. 2016. The Computational Power of Optimization in Online Learning. In *Proceedings of the 48th Annual ACM Symposium on Theory of Computing (STOC)*. ACM, 128–141.

[31] Elad Hazan and Tomer Koren. 2016. Learning in Games with Best-Response Oracles (Presenation). https://simons.berkeley.edu/sites/default/files/docs/5595/koren.pdf

[32] Elad Hazan, Tomer Koren, Roi Livni, and Yishay Mansour. 2016. Online Learning with Low Rank Experts. In *Proceedings of the 29th Conference on Learning Theory, COLT*. 1096–1114.

[33] Zhiyi Huang, Yishay Mansour, and Tim Roughgarden. 2018. Making the most of your samples. *SIAM J. Comput.* 47, 3 (2018), 651–674.

[34] Marcus Hutter and Jan Poland. 2005. Adaptive online prediction by following the perturbed leader. *Journal of Machine Learning Research (JMLR)* 6 (2005), 639–660.

[35] Sham Kakade and Adam Tauman Kalai. 2005. From Batch to Transductive Online Learning. In *Proceedings of the 18th Annual Conference on Neural Information Processing Systems (NeurIPS)*. 611–618.

[36] Sham M Kakade, Adam Tauman Kalai, and Katrina Ligett. 2009. Playing games with approximation algorithms. *SIAM J. Comput.* 39, 3 (2009), 1088–1106.

[37] Adam Kalai and Santosh Vempala. 2005. Efficient algorithms for online decision problems. *J. Comput. System Sci.* 71, 3 (2005), 291–307.

[38] Jyrki Kivinen and Manfred K. Warmuth. 1997. Exponentiated Gradient versus Gradient Descent for Linear Predictors. *Information and Computation* 132, 1 (1997), 1–63.

[39] Robert Kleinberg and Tom Leighton. 2003. The value of knowing a demand curve: Bounds on regret for online posted-price auctions. In *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. IEEE, 594–605.

[40] John Langford and Tong Zhang. 2008. The epoch-greedy algorithm for multi-armed bandits with side information. In *Proceedings of the 22nd Annual Conference on Neural Information Processing Systems (NeurIPS)*. 817–824.

[41] Nick Littlestone and Manfred Warmuth. 1986. *Relating data compression and learnability.* Technical Report. Technical report, University of California, Santa Cruz.
[42] Nick Littlestone and Manfred K Warmuth. 1989. The weighted majority algorithm. In *Proceedings of the 30th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. IEEE, 256–261.
[43] Jamie H Morgenstern and Tim Roughgarden. 2015. On the pseudo-dimension of nearly optimal auctions. In *Proceedings of the 29th Annual Conference on Neural Information Processing Systems (NeurIPS)*. 136–144.
[44] Roger B Myerson. 1981. Optimal auction design. *Mathematics of operations research* 6, 1 (1981), 58–73.
[45] Noam Nisan and Amir Ronen. 2007. Computationally Feasible VCG Mechanisms. *Journal of Artifificial Intellegience Research (JAIR)* 29, 1 (May 2007), 19–47.
[46] Daniel Paulin. 2015. Concentration inequalities for Markov chains by Marton couplings and spectral methods. *Electronic Journal of Probability* 20 (2015).
[47] David Pollard. 2012. *Convergence of stochastic processes.* Springer Science & Business Media.
[48] Alexander Rakhlin and Karthik Sridharan. 2016. BISTRO: An Efficient Relaxation-Based Method for Contextual Bandits. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*. 1977–1985.
[49] Tim Roughgarden and Okke Schrijvers. 2016. Ironing in the Dark. In *Proceedings of the 17th ACM Conference on Economics and Computation (EC)*. ACM, 1–18.
[50] Tim Roughgarden and Joshua R Wang. 2016. Minimizing Regret with Multiple Reserves. In *Proceedings of the 17th ACM Conference on Economics and Computation (EC)*. ACM, 601–616.
[51] Tuomas Sandholm, Andrew Gilpin, and Vincent Conitzer. 2005. Mixed-integer programming methods for finding Nash equilibria. In *Proceedings of the 20th AAAI Conference on Artificial Intelligence (AAAI)*. 495–501.
[52] Tuomas Sandholm, Subhash Suri, Andrew Gilpin, and David Levine. 2002. Winner determination in combinatorial auction generalizations. In *Proceedings of the 1st International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*. 69–76.
[53] Vasilis Syrgkanis, Akshay Krishnamurthy, and Robert E. Schapire. 2016. Efficient Algorithms for Adversarial Contextual Learning. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*. 2159–2168.
[54] Vasilis Syrgkanis, Haipeng Luo, Akshay Krishnamurthy, and Robert E Schapire. 2016. Improved Regret Bounds for Oracle-Based Adversarial Contextual Bandits. In *Proceedings of the 30th Annual Conference on Neural Information Processing Systems (NeurIPS)*. 3135–3143.