Learning What's Going on: Reconstructing Preferences and Priorities from Opaque Transactions

AVRIM BLUM, Carnegie Mellon University
YISHAY MANSOUR, Tel Aviv University
JAMIE MORGENSTERN, Carnegie Mellon University

We consider a setting where n buyers, with combinatorial preferences over m items, and a seller, running a priority-based allocation mechanism, repeatedly interact. Our goal, from observing limited information about the results of these interactions, is to reconstruct both the preferences of the buyers and the mechanism of the seller. More specifically, we consider an online setting where at each stage, a subset of the buyers arrive and are allocated items, according to some unknown priority that the seller has among the buyers. Our learning algorithm observes only which buyers arrive and the allocation produced (or some function of the allocation, such as just which buyers received positive utility and which did not), and its goal is to predict the outcome for future subsets of buyers. For this task, the learning algorithm needs to reconstruct both the priority among the buyers and the preferences of each buyer. We derive mistake bound algorithms for additive, unit-demand and single minded buyers. We also consider the case where buyers' utilities for a fixed bundle can change between stages due to different (observed) prices. Our algorithms are efficient both in computation time and in the maximum number of mistakes (both polynomial in the number of buyers and items).

Categories and Subject Descriptors: F.2.0 [**Theory of Computation**]: ANALYSIS OF ALGORITHMS AND PROBLEM COMPLEXITY

General Terms: Mechanism Design, Learning Theory, Algorithms

Additional Key Words and Phrases: Mechanism design; mistake-bound learning

ACM Reference Format:
Avrim Blum, Yishay Mansour, Jamie Morgenstern, 2015. Learning What's Going on: Reconstructing Preferences and Priorities from Opaque Transactions. ACM X, X, Article X (February 2015), 19 pages.
DOI = 10.1145/2764468.2764492 http://doi.acm.org/10.1145/2764468.2764492

1. INTRODUCTION

A collection of lobbyists enter a politician's office. An hour later, they emerge, some happy and some unhappy. The next day, a different subset of lobbyists enter, and again some emerge happy and some unhappy. Suppose that what is happening is that the politician has a collection of m favors (items) to distribute, along with a priority ordering over lobbyists; the lobbyists are single-minded, each lobbyist i with a demand-set $D_i \subseteq \{1, \ldots, m\}$. The politician orders the lobbyists who arrived that day by priority

Blum was supported in part by the National Science Foundation under grants CCF-1101215, CCF-1116892, CCF-1331175, and IIS-1065251. Email: avrim@cs.cmu.edu. Mansour was supported in part by The Israeli Centers of Research Excellence (I-CORE) program, (Center No. 4/11), by a grant from the Israel Science Foundation (ISF), by a grant from United States-Israel Binational Science Foundation (BSF), and by a grant from the Israeli Ministry of Science (MoS). Email: mansour@tau.ac.il, Morgenstern was supported in part by the National Science Foundation under grants CCF-1116892, CCF-1331175, CCF-1415460 and by a Simons Award for Graduate Students in Theoretical Computer Science. Email: jamiemmt@cs.cmu.edu.

EC'15, June 15–19, 2015, Portland, OR, USA. ACM 978-1-4503-3410-5/15/06 ...$15.00.

http://dx.doi.org/10.1145/2764468.2764492

and hands each one D_i if it is still available (making the lobbyist happy) or handing her nothing if D_i is no longer available (making the lobbyist unhappy). Can we reconstruct the politician's priority ordering and the lobbyists' demand-sets (or at least, given a set of lobbyists, predict which will end up happy and which unhappy, since we cannot observe the items themselves) from these types of observations?

Alternatively, in the context of computational advertising, consider a publisher that owns a web site and has some collection of advertisers. The advertisers tell the publisher which potential impressions are relevant to them (are they interested in this type of impression), their bid (the value they are willing to pay for a relevant impression) and conflicts (which competing advertisers they refuse to appear concurrently with, for example, competing car makers for a car ad). Each time a user visits a webpage, the publisher's ad server considers the subset of relevant advertisers. It then orders the advertisers (say, by their bid) and greedily assigns an impression to an advertiser if it does not introduce a conflict (otherwise it skips this advertiser). From observing which advertisements are shown and which are not, and knowing which advertisers are relevant, can we learn the conflicts and priority ordering?

In this paper, we consider this and several closely related problems. Formally, we assume there are n buyers (lobbyists or advertisers) and a mechanism (the politician or ad server) which has a priority ordering over buyers that is unknown to us. There is a collection of m items, and the buyers each have utility functions over subsets of items (e.g., the examples above correspond to the case of single-minded buyers[1]). At each time-step t, some set $S^t \subseteq \{1, \ldots, n\}$ of buyers arrive. The mechanism then orders the buyers in S^t by priority and allocates to each its most-preferred bundle from the collection of items not yet given to earlier buyers in the ordering. Finally, we observe some function y^t of the outcome (allocation). We will consider the case that buyers are single-minded and y^t indicates which buyers received positive utility and which did not (as in the examples above), as well as the case that buyers are unit-demand or additive, and y^t indicates the items (if any) that each buyer received. Our goal will be to predict y^t from S^t, and we will present efficient algorithms that can do so while making only a bounded (polynomial in n and m) number of mistakes in total. Formally, we will be working in the *online mistake-bound model*, where on each round t we are given S^t and try to predict y^t. If our prediction is incorrect we are shown the true y^t and charged one *mistake*. Our goal is to design polynomial-time algorithms that have a worst-case number of mistakes (known as mistake bounds) that are as small as possible. We remark that we consider priority-based mechanisms because they are particularly clean and have the desirable property that allocation is easy when all the information is known. It would be of interest to consider this problem for other kinds of mechanisms as well.

Notice that the setting of single-minded buyers can exhibit significant non-monotonicities. For example, consider two lobbyists a and b whose demand sets D_a and D_b do not overlap, and with a having higher priority than b. It could be that a's presence has no effect on b (since their sets don't overlap); it could be that a's presence *helps* b (if there is a lobbyist c present, with priority between a and b, such that $D_a \cap D_c \neq \emptyset$ and $D_c \cap D_b \neq \emptyset$); or, it could be that a's presence *hurts* b (if there are lobbyists d, e present with $D_a \cap D_d \neq \emptyset$, $D_d \cap D_e \neq \emptyset$, and $D_e \cap D_b \neq \emptyset$, with ordering $a \succ d \succ e \succ b$).

To get a feel for the type of results we are aiming for, we describe here a simpler case of this problem and how one can solve it (and, for an overview of the results, see

[1]In the case of advertisers, each pairwise conflict can be modeled as an abstract item that belongs to the demand-set of both conflicting advertisers.

Table 1). Suppose buyers are additive rather than single-minded,[2] and y^t denotes the *allocation* of items to the agents in S^t. This problem is monotonic in some sense: if $S^{t'} \subseteq S^t$ and $i \in S^{t'}$, then $y_i^t \subseteq y_i^{t'}$ (including more buyers reduces the allocation for i). It is possible to solve this problem tracking two things: first, for a given buyer i, track the set of items i has ever won, and second, an estimate the relative ordering of the buyers $\widehat{\succ}$. Consider some item j that buyer i wins in some round. Buyer i will take item j whenever it is still available, so if buyer i doesn't win item j, we learn that buyer i is later in the ordering than the winner of item j. To predict the allocation for a set S^t, we order buyers in S^t according to $\widehat{\succ}$, and in that order, give the buyers all of the remaining items she has bought before.

The algorithm will make two types of mistakes, and we can limit the number of each. Consider the first buyer (according to $\widehat{\succ}$) for whom we make a mistake in predicting her allocation. Suppose she won some item j that we did not predict she would get. Since she was the first mistake according to $\widehat{\succ}$, we did not predict that someone earlier in $\widehat{\succ}$ won item j. Thus, item j was available in our prediction when we reached buyer i: we did not allocate item j to her because we had never seen her win item j before. There are at most nm of these sorts of mistakes to make (one per item/buyer pair). Suppose instead we predicted that some item j would be allocated to buyer i but she did not get item j. Since we predicted item j for buyer i, buyer i must have won item j before (and is therefore interested in item j). Then, it must be the case that buyer i is later in the true ordering than in $\widehat{\succ}$: the winner of item j must be earlier than buyer i. Then, we can update $\widehat{\succ}$ by demoting buyer i: if done carefully, as we describe in Section 3, i will never be demoted further than her true position in the ordering, so there will be at most n^2 mistakes of this type.

1.1. Our Results

This paper presents several mistake-bound learning algorithms for priority-ordered mechanisms. The crux of these algorithms is to learn the hidden *priority order*, or permutation over buyers, in a way that meshes well with learning the players' preferences at the same time, all in a mistake-bound framework. First, we consider the case without prices (or equivalently, when prices are fixed across time). In the case of a single item, this problem reduces to learning the priority order over buyers. Previous work describes how to efficiently sample linear extensions of partial orders [Karzanov and Khachiyan 1991], which can be combined with a simple halving algorithm to learn a permutation with a mistake bound of $\Theta(n \log(n))$ when mistakes are accompanied with some pair i, j which were mis-ordered (see Section 6). When buyers have more general valuations, however, it is not clear how to use this algorithm to learn the priority order over buyers.[3] algorithm for learning a permutation whose mistake bound is $\Theta(n^2)$ (when a mistake is accompanied by some element of the permutation that needs to be demoted rather than a pair of elements for which the permutation was incorrect). With this algorithm, we build mistake-bound learning algorithms for single-minded, unit-demand, and additive buyers with fixed prices, and for unit-demand and additive buyers with variable, observable prices. The precise form of these bounds is summarized in Table 1, along with several information-theoretic lower bounds. The results for

[2]For each buyer i and each item j, either buyer i either wants j or she doesn't, and buyer i takes all items that she wants that are available when it is her turn. One can think of this as the behavior of additive buyers in the presence of fixed prices.

[3]In the case of a single item, we learn that the true winner has higher priority than everyone else. In general, mistakes don't give such a simple constraint on the ordering of buyers. When there are multiple items, a buyer may get item a or b, depending upon who else shows up: one of these might be due to her preferences, and one might have to do with the availability of one of the items; it is not a priori clear which is the case.

Type of buyers	Prices (fixed or variable)	Mistake Bound
Single-minded	fixed	$\Theta(n^2)$
Single-item	fixed	$\Theta(nm\log(m))$
Additive	fixed	$O(nm+n^2)$
Unit-demand	fixed	$O(n^2m\log(m)), \Omega(nm\log(m))$
Additive	variable	$O(n^2m\log(V))$
Unit-demand	variable	$O(n^2\mathcal{MB})$

Fig. 1. Summary of our results; V is the maximum value any buyer has for an item and $\mathcal{MB}$ is the mistake bound for the Ellipsoid algorithm. The lower bounds are information-theoretic. Our algorithm for single-minded buyers applies even to the case where observations are only which buyers get their set and which do not, rather than the explicit allocation.

additive and unit-demand buyers also apply to the case where there are multiple copies of goods. Our results for single-minded buyers hold for a restricted model where our observations (and goal for prediction) are only the set of satisfied agents, rather than the allocation. We also show that this restricted observation model is computationally intractable for *unit-demand* bidders under cryptographic assumptions by reducing to the problem of learning boolean formulas [Kearns and Valiant 1994].

1.2. Related Work

Our work is related to the literature on learning from revealed preferences [Samuelson 1938][Varian 2006], which considers the problem of learning about a single buyer from observing her behavior under observed prices. Rationalizable demands (according to some prices) are those which can arise from maximizing some concave, monotone, continuous value function subject to a budget constraint. Beigman and Vohra [2006] show that there exist continuous, nondecreasing and concave utility functions which have unbounded query complexity; they give algorithms with finite sample complexity bounds for learning and predicting from rationalizable demand/price pairs for some special cases. Zadimoghaddam and Roth [2012] gave computationally efficient versions of these algorithms for linear and linearly separable utilities. Amin et al. [2014] consider both the problem of setting prices (minimizing regret w.r.t revenue) and the prediction problem (minimizing classification mistakes w.r.t. exogenous prices) for the online version of this problem. Balcan et al. [2014] give tight sample complexity results for predicting linear, separable piecewise linear concave, CES, and Leontif preferences, and extend the results to the agnostic setting. Their results extend to indivisible goods and certain nonlinear pricing.

Our work differs from the work on revealed preferences in two key ways. One is that in the settings we consider, each observed transaction involves multiple buyers: as a result, a buyer might not get some item because it was not in her desired bundle, or might not get it because the item (or a complement to it) was taken by some other buyer. The other is that we also, at the same time, are aiming to learn an unknown priority ordering for the seller. The algorithms we derive use as subroutines algorithms that are closely related to those for learning decision lists in a mistake bound setting [Helmbold et al. 1990]. We also employ a mistake bound learner for classification by halfspaces from Maass and Turán [1990] for efficiently learning a classifier for a fixed unit-demand buyer.

2. MODEL AND PRELIMINARIES

Let B be a set of n buyers and I be a set of m items. Each buyer $i \in B$ has some combinatorial valuation v_i over subsets of items $T \subseteq I$. A *priority-ordered mechanism* $\mathcal{M}$ consists of an ordering over buyers $\succ$ and allocates items in I as follows. At each

time t, an arbitrary subset S^t of the n buyers arrives online. Then, in order according to $\succ$, each buyer in S^t chooses the bundle from the remaining items that maximizes her value. So, the buyer i_1 in S^t who is first according to $\succ$ chooses the bundle $X^t_{i_1} \subseteq I$ of maximum value to her, then the buyer i_2 in S^t who is second according to $\succ$ chooses the bundle $X^t_{i_2} \subseteq I \setminus X^t_{i_1}$ of maximum value to her, and so on.

The *label* y^t for the example S^t is some function $\texttt{obs}$ of the allocation $(X^t_1, \ldots, X^t_n) = \mathcal{M}(S^t)$ which arises from this process. Our goal will be to predict $y^t = \texttt{obs}(\mathcal{M}(S^t))$ for a new subset S^t given our previous observations. We will focus on $\texttt{obs} = Id$ (the identity function; our goal is to predict the allocation) and $\texttt{obs} = (\mathbb{I}[X^t_1 \neq \emptyset], \ldots, \mathbb{I}[X^t_n \neq \emptyset])$ (the function which indicates which buyers bought at least one item; our goal is to predict which players have positive utility).

We will be working in the **mistake-bound model**, where our learning algorithm $\mathcal{M}$ will progress as follows. In each round, the algorithm is presented with a subset S^t. The algorithm's current prediction, $\mathcal{M}(S^t)$, will be output. Then, the algorithm observes the true label $y^t = \texttt{obs}(X^t_1, \ldots, X^t_n)$. If $\mathcal{M}(S^t) \neq y^t$ (the label predicted is incorrect), round t is counted as a **mistake**. The goal in mistake-bound learning is to bound the worst-case total number of mistakes made over an arbitrarily long sequence of examples presented to the algorithm. We will call this worst-case bound the **mistake bound for learning algorithm** $\mathcal{M}$.

We also consider an extension of this setting, where each example is a pair (S^t, p^t) of a subset of buyers and a price vector $p^t \in \mathbb{R}^m$. Then, in order according to $\succ$, the buyers in S^t each chooses a bundle from the remaining items maximizing her *utility*. We assume that their utility is *quasi-linear in money*, i.e., that $u_i(X^t_i, p^t) = v_i(X^t_i) - \sum_{j \in X^t_i} p^t(j)$ (where $X^t_i \subseteq I$ represents the bundle player i chose). We call this model the *variable-price model*; the previous model, described without prices, can be thought with a fixed price vector p which does not vary across examples. We also note that, for several of the problems we study, our algorithms extend to the case where for each item $e \in I$, there are k_e *copies* of that item (here, we assume buyers are unit-demand in each item, so they will never purchase more than one copy of a given item).

3. SINGLE-MINDED BUYERS

Suppose there are n single-minded buyers. That is, each bidder i has a single demand set D_i for which she has value $v_i(D_i) > 0$. At each time t, a subset $S^t \subseteq [n]$ arrives, and each buyer in S^t is offered the items which haven't been taken by earlier buyers. If buyer i is offered some set which includes D_i, she will take D_i, otherwise she will take nothing (one can imagine ϵ prices giving a disincentive to take excess items). Let W^t denote the set of buyers who won their demand sets in example S^t (so $W^t = \texttt{obs}(\mathcal{M}(S^t))$). From W^t and S^t alone, we wish to be able to predict the winning set $W^{t'}$ for a new subset $S^{t'}$, where our performance objective is the number of mistakes made.

Suppose for a moment our learning algorithm knew $\succ$ but not the demand sets D_i. From $\succ$ alone, we cannot predict sets $W^{t'}$ given $S^{t'}$: we need to understand which buyers have items in common between their demand sets (e.g., if $D_i \cap D_j \neq \emptyset$, then i and j will not simultaneously be in any winning set). For this reason, we call i and j *in conflict* if $D_i \cap D_j \neq \emptyset$. Consider the graph $G(V, E)$ where $V = [n]$ and there is an edge $(i, j) \in E$ iff buyers i and j have a conflict: we call this the *conflict graph*. For each pair of buyers (i, j), if $i, j \in W^t$ for some S^t, the two buyers clearly do not have a conflict. Notice that, even if $i \succ j$, there will be cases when $j \in W^t$ but $i \notin W^t$ (if i has a conflict, say, with an earlier winning buyer, but j does not). Thus, to predict whether buyer i will win, it is not sufficient to know simply who has precedence over i and who conflicts with i: we need more information about the structure of the conflict graph.

Notice that, given the conflict graph G and the ordering over buyers $\succ$, we can predict the winners W^t for an arbitrary subset S^t. In particular, buyer i wins exactly when no one before i won with whom i has a conflict. So, we can predict the entire winning set by scanning through S^t in the order $\succ$ and adding buyer i to W^t if no conflicting buyer is already in W^t. Thus, we are done if we can learn both G and $\succ$.

We start by presenting a mistake-bound procedure PermULearn (informally alluded to in the introduction) to learn a permutation $\succ$ in a model where each time it makes a mistake by guessing $\widehat{\succ}^t$, it is told some item d^t that is incorrectly above some other item r in $\widehat{\succ}^t$ (but is not told r). The procedure employs a data structure $P = (\tilde{\succ}, O)$, where $\tilde{\succ}$ is a permutation and O is a partition of the buyers to levels. Given a permutation σ, let $\texttt{Loc}(i, \sigma)$ denote the position of i in σ and let $\texttt{Buyer}(k, S, \sigma)$ denote the name of the kth element in σ restricted to S. The data structure at time t is P^t and we will associate P^t with its output permutation $\widehat{\succ}^t$, so that $\texttt{Loc}(i, P^t) = \texttt{Loc}(i, \widehat{\succ}^t)$. The key interface with P^t will be the function $\texttt{Demote}(d^t, P^t)$, employed when d^t is returned as a mistake (an item that is incorrectly above some other item r in $\widehat{\succ}^t$). The algorithm is quite similar to that used for learning decision lists in a mistake bound setting [Helmbold et al. 1990]. We use the following lemma throughout the rest of our analysis.

Subroutine PermULearn: Maintains level ordering over $[n]$ and predicts permutation $\widehat{\succ}^t$, an estimate of $\succ$. Given an error, receives $i \in [n]$ s.t. $\exists r$ s.t. $i \widehat{\succ}^t r$ but $r \succ i$.

Let $\tilde{\succ}$ be an arbitrary ordering for tie breaking;
* `InitPerm`, put all items at level 1 *
$O_1 = \{1, \ldots, n\}$ $\quad O_{2,\ldots,n} = \emptyset$ $\quad \widehat{\succ}^0 = \tilde{\succ}$;
* $\texttt{Permutation}(P^t)$, outputs consistent permutation *
for $l = 1$ *to* $|O|$ **do**
  π_l = Order O_l, level l of O, according to $\tilde{\succ}$;
$\widehat{\succ}^t = \pi_1 \cdot \pi_2 \cdot \ldots \pi_l$ the concatenation of the levels' permutations;
* $\texttt{Demote}(i, P^t)$, demote bidder i *
Let k be the level in which i resides, e.g., $i \in O_k$;
Let $O'_k = O_k \setminus \{i\}, O'_{k+1} = O_{k+1} \cup \{i\}$;
Let $O' = (O_1, \ldots, O_{k-1}, O'_k, O'_{k+1}, O_{k+1}, \ldots)$;
Let $P^{t+1} = (\tilde{\succ}, O')$;
$\widehat{\succ}^{t+1} = \texttt{Permutation}(P^{t+1})$;

LEMMA 3.1. *PermULearn has a mistake-bound of $O(n^2)$ for learning a permutation $\succ$ if when $\widehat{\succ}^t \not\succ$, it is given some $d^t \in [n]$ such that $\exists r$ such that $d^t \widehat{\succ}^t r$ but $r \succ d$.*

PROOF. Let $\texttt{Level}(i, O)$ be the level of buyer i in O, i.e., $\texttt{Level}(i, O) = j$ where $i \in O_j$. Let d^t be the element given (and demoted by PermULearn) at time t. We show by induction on the algorithm's pushing down elements that no d^t is pushed to a level below her location in $\succ$, i.e., $\texttt{Loc}(i, \succ) \geq \texttt{Level}(i, O^t)$. If this is the case, the algorithm is correct: at most n^2 pushes can occur and the limit of these push-downs is some consistent permutation. Prior to any elements being pushed down, all elements are at the first level, hence $\texttt{Loc}(i, \succ) \geq \texttt{Level}(i, O^0)$ initially.

Now, suppose that the inductive hypothesis holds at time $t-1$, i.e., for any item i we have $\texttt{Loc}(i, \succ) \geq \texttt{Level}(i, O^{t-1})$. Our induction hypothesis implies two things: first, that $\texttt{Loc}(d^t, \succ) \geq \texttt{Level}(d^t, O^{t-1})$ and second, $\texttt{Loc}(r, \succ) \geq \texttt{Level}(r, O^{t-1})$

Our assumption states that when d^t is demoted, there is some element r such that $\texttt{Loc}(d^t, \succ) > \texttt{Loc}(r, \succ)$ but $\texttt{Loc}(r, P^t) > \texttt{Loc}(d^t, P^t) \geq \texttt{Level}(d^t, O^{t-1})$. The second implication of our induction hypothesis states, prior to the t-th demotion, $\texttt{Loc}(r, \succ) \geq \texttt{Level}(r, O^{t-1})$. Since $\texttt{Loc}(r, P^t) > \texttt{Loc}(d^t, P^t)$, it must be the case that $\texttt{Level}(r, O^{t-1}) \geq \texttt{Level}(d^t, O^{t-1})$ (r is only given a later location in P^t than d^t if she is at a weakly larger-numbered level). Thus, $\texttt{Loc}(d^t, \succ) > \texttt{Loc}(r, \succ) \geq \texttt{Level}(r, O^{t-1}) \geq \texttt{Level}(d^t, O^{t-1})$, so $\texttt{Loc}(d^t, \succ) > \texttt{Level}(d^t, O^{t-1})$ and pushing d^t to level $\texttt{Level}(d^t, O^{t-1}) + 1$ maintains the invariant. So the algorithm is correct, and no element is pushed more than n times, implying a mistake bound of n^2. □

Thus, PermULearn is guaranteed to learn a consistent ordering, so long as we never request a demotion of a buyer who shouldn't be demoted. So, when we use this procedure, it suffices to show that we never demote a buyer i unless there is some buyer i' later in P^t but earlier in $\succ$ to guarantee that we learn a consistent ordering with at most $O(n^2)$ mistakes.

Before giving our main result, we first briefly mention that PermULearn is enough to learn in the simpler case that $\texttt{obs} = Id$ (we observe the entire allocation, not just who is in the winning set).

COROLLARY 3.2. *There is an efficient algorithm for predicting when* $\texttt{obs} = Id$ *(the allocation) for subsets of single-minded bidders with a mistake bound of* $O(n^2)$.

PROOF. Use an instantiation of the permutation data structure P as above. For each bidder i, let $\hat{D}_i = \emptyset$ initially. Whenever we see a bidder i win a nonempty set, set $\hat{D}_i = X_i = D_i$. When a subset S^t arrives, in order according to P, allocate $j \in S^t$ his set $\hat{D}_j$ if it is still available (otherwise, $\hat{X}_j = \emptyset$). When this allocation rule makes a mistake, consider i^t, the first bidder (according to the ordering P) for which our algorithm mis-allocated items. There are two possible mistakes: $\hat{X}_{i^t} = \emptyset$ but $X_{i^t} = D_{i^t}$, or $X_{i^t} = \emptyset$ but $\hat{X}_{i^t} = D_{i^t}$. The first case can occur for two reasons: we have never seen i^t win, in which case we have $\hat{D}_{i^t} = \emptyset$, so we then set $\hat{D}_{i^t} = D_{i^t}$ (there are at most n of these errors), or because there is some i' such that $i^t \succ i'$ but $i' \widehat{\succ}^t i^t$ (who conflicts with i^t). But this is not possible, or i' would be an earlier mistake. The second kind of mistake can only occur because there is some i' such that $i' \succ i^t$ but $i^t \widehat{\succ}^t i'$ (i^t is too early in the permutation). Thus, by Lemma 3.1, demoting i^t in these cases is valid and leads to a mistake bound of $O(n^2)$. Thus, in total, there are $O(n^2)$ mistakes made by this algorithm. □

It remains to show how we can use the permutation data structure to solve our original problem, that of learning $\succ$ alongside the conflict graph for single-minded buyers when we only observe the winning sets W^t. The intuition behind our main algorithm is as follows. We will initialize the permutation data structure and begin by assuming the conflict graph is the complete graph. For a given estimate $\widehat{\succ}$ and conflict graph $\widehat{G}$, we predict $\widehat{W^t}$ for S^t as follows. The mechanism serves members of S^t in order according to $\widehat{\succ}$ (e.g, $\texttt{Loc}(i, P^t) < \texttt{Loc}(j, P^t)$ will imply i gets served before j), subject to the constraint that if j is in conflict with some earlier buyer who has won, j doesn't win. Then, there will be two types of mistakes: when W^t includes some pair (i, j) connected by an edge in $\widehat{G}$, and when it does not. In the first case, we can safely remove (i, j) from $\widehat{G}$, and in the second case we will argue that we can safely demote some buyer. We will

maintain the invariants alluded to previously: namely, that $E \subseteq \widehat{E}$ (for edges in the conflict/current estimate graph), and that we have never demoted a buyer who didn't need to be demoted. Algorithm SingleMinded formalizes this intuition.

Algorithm SingleMinded: MB algorithm for predicting winners; single-minded buyers wrt order $\succ$

P=`InitPerm`;
Let $\widehat{G} = ([n], \widehat{E})$ where $(i,j) \in \widehat{E}$ for all $i \neq j$;
for $t = 1$ *to* T **do**
  Receive S^t;
  Let $\widehat{W}^t = \emptyset$;
  for $b = 1$ *to* $|S^t|$ **do**
    Let $i = \texttt{Buyer}(b, S^t, P)$;
    add i to $\widehat{W}^t$ if $\nexists j \in \widehat{W}^t$ such that $(i,j) \in \widehat{E}$;
  Predict $\widehat{W}^t$;
  Learn W^t;
  if $W^t \neq \widehat{W}^t$ **then**
    if $\exists i, j \in W^t$ *such that* $(i,j) \in \widehat{E}$ **then**
      $\widehat{E} = \widehat{E} \setminus \{(i,j)\}$
    else
      Let $i^t = \texttt{Buyer}(1, \widehat{W}^t \setminus W^t, P)$;
      $P = \texttt{Demote}(i^t, P)$;

THM 3.1. *SingleMinded is a $2n^2$-mistake bound algorithm for predicting W^t, the winning set for single-minded buyers.*

PROOF. Throughout the life of the algorithm, two invariants are maintained. First, the true set of edges in the conflict graph E will always be contained in $\widehat{E}$ the set of conflicts the algorithm tracks. Second, $\widehat{\succ}$ will only be told to push down a buyer i when there is some j such that $\texttt{Loc}(i, \succ) > \texttt{Loc}(j, \succ)$ but $\texttt{Loc}(i, P^t) < \texttt{Loc}(j, P^t)$.

We proceed to show the first invariant holds. It begins with the complete conflict graph $\widehat{G} = ([n], \widehat{E})$, so the invariant holds at the beginning. Whenever the algorithm deletes an edge (i,j) from $\widehat{G}$, an example has been observed where two buyers are clearly not in conflict (e.g., i and j are both allocated in some example).

Now, we prove the second invariant. This is clearly true when we push some i down the first time; i doesn't always win when he shows up, implying he isn't at the first level according to $\succ$. Now, suppose so far this has been the case: no buyer so far has been pushed unless there was proof according to $\succ$ that he is below someone below him according to $\widehat{\succ}^{t-1}$. Then, when i^t is asked to be pushed down, it is because i^t wasn't allocated to, even though $\widehat{\succ}^{t-1}$ said he should have been. This isn't because of conflicts, by invariant 1, so i^t didn't conflict with those above him according to $\widehat{\succ}^{t-1}$. Moreover, since i^t was *the first person according to $\widehat{\succ}^{t-1}$ where we made a mistake*, i^t must conflict with someone above him according to $\succ$, implying there is someone below him in $\widehat{\succ}^{t-1}$ who he is below in $\succ$. Thus, the invariant is maintained after i^t is demoted.

Thus, by Lemma 3.1, using the permutation data structure is appropriate: it is never told to push down some buyer who doesn't need to be lower according to $\succ$, so the algorithm is correct. Finally, there can be at most n^2 edges deleted from the conflict graph, and at most n^2 times where some buyer is pushed down in the ordering. Thus, the mistake bound on this algorithm is $2n^2$. □

Thus, this is quite an effective way to interpret the observations of "satisfied" or "not satisfied": if the observation was actually the allocation (and the goal to predict the allocation), the problem trivially reduces to the n^2 bound from the single-item case (it reduces to learning the priority of each buyer and seeing each buyer win one time). If, on the other hand, the observations are simply the subset of those buyers who are satisfied, and we aim to construct the smallest consistent model of the items corresponding to the demanded sets, the problem becomes NP-complete.

OBSERVATION 3.2. *Given a conflict graph G, finding the smallest m for which single-minded bidders over m items suffices to describe the conflicts in G is equivalent to clique edge-cover, and is thus NP-complete. On the other hand, there will always exist a consistent set of at most n^2 items: in particular, one item for each edge in G with each player wanting all of its incident edges.*

4. UNIT-DEMAND BUYERS

Suppose now our n buyers are unit-demand, and we wish to predict the allocation rather than just the winning set. When prices are fixed, this problem corresponds to each buyer having an ordering over items $\succ_i$, as well as the ordering $\succ$ over buyers. At each time t, a subset S^t arrives, and the players in S^t, in order according to $\succ$, will each choose their favorite item remaining. For example, the first buyer in S^t will choose his favorite item, the second buyer will choose his favorite that the first buyer didn't take, and so on. By a reduction to the single-minded case, we have the following.

THM 4.1. *There is an efficient $O(n^2m^2)$-mistake bound learning algorithm for predicting the allocation for subsets of buyers, according to an priority-ordered mechanism with fixed prices and unit-demand buyers, when observations are true allocations.*

PROOF. For each player i, make m "ghost" buyers $i_1 \dots i_m$, one for each item j, which will correspond to embedding i's preferences into $\succ$. The true allocation mechanism $\mathcal{M}$ can be viewed as an ordering of the mn ghost buyers (an ordering over buyers, and within each buyer an ordering over items) where two ghost buyers are in conflict if either they correspond to the same buyer or correspond to the same item. Now, consider Algorithm SingleMinded. Since only one of the m copies of a given buyer will be allocated to according to $\mathcal{M}$, we will never delete conflict edges between these copies, and will thus never predict two ghosts corresponding to the same true buyer will win simultaneously. Similarly, for any item j, since the item will never simultaneously be given to two buyers, we will never delete conflict edges corresponding to that item. Finally, for a player in position j according to $\succ$, no more than j of his ghost buyers will ever be seen winning. Thus, at most j of his ghosts will need to be rearranged by $\hat{\succ}$. In total, then, there are only $\min(n^2, nm)$ ghost buyers that are relevant. Then, there are at most $O(\min(n^4, n^2m^2))$ mistakes, by Theorem 6.2. □

We also briefly mention that we have a slightly tighter bound, whose proofs are relegated to Section A.1.

THM 4.2. *Algorithm UnitDemandPrime has a mistake bound of $\Theta(n^2 m \log(m))$ for predicting allocations for subsets of unit-demand buyers, when the $\mathcal{M}$ is a priority-ordered mechanism with fixed prices when the observation is the true allocation.*

4.1. $k_{i,t}$-demand

The previous results extend to cases where bidder i wishes to buy k_i items, rather than just 1, and chooses the top k_i with respect to her ordering of the remaining available items, even if the k_is are not known in advance. The algorithm needs to maintain a conservative estimate of each k_i, so the initial estimate should be $\hat{k}_i = 0$ for all i. As before, when we make a mistake, we will only update the *first* buyer (according to our internal ordering over buyers) for whom our algorithm incorrectly predicted T rather than S. There will now be 2 types of mistakes: where an agent buys a set S which is superset of T, the $\hat{k}_i$ items we predicted (in which case our algorithm should set $\hat{k}_i = |S|$), and when an agent buys a set S when we predicted T and $T \setminus S \neq \emptyset$ (in which case, we demote the *first* item in $T \setminus S$ according to our internal ordering over i's items). The total mistake bound will increase by $\sum_i k_i$. Similarly, if at each time t, agent i desires $k_{i,t}$ items, and $k_{i,t}$ is part of the *input*, predicting that agent i will buy her $k_{i,t}$ favorite items, with the same update rule, will yield the same mistake bound as in the usual unit-demand case.

4.2. Multiple copies

Several of these results are easy to extend to the setting where there are multiple copies of each resource, and players are unit-demand for multiple copies of a particular item (e.g., no player wants more than one copy of a given item). If buyers are additive (across bundles of different items), it suffices to learn their preferences over item *types*. This can be done with no loss, treating any copy of a resource identically internal to the learning algorithm. For prediction, a buyer will take his favorite *bundle* of items, and that bundle can contain an item for which there is at least one copy remaining. This implies a mistake bound for learning the allocations which is *independent* of the number of copies of each item.

For unit-demand buyers, it is not clear how to use the solution from Theorem 4.1, which reduces to the single-minded case (for buyer i to not have item e available, there would need to be k_e "ghosts" that bought item e prior to buyer i, rather than a single conflict). However, Algorithm UnitDemandPrime can be used directly to learn the permutation over buyers and each buyer i's preference order over item types.

In the case of single-minded buyers, recall that the problem is quite easy if we ever see an allocation; Corollary 3.2 applies directly with no loss in the mistake bound. On the other hand, if no allocation is seen, and instead we only see the subset of people who received their set, one can use a *conflict hypergraph* rather than a conflict graph. The total number of edges in a necessary hypergraph blows up rather quickly, unfortunately: the size of this representation (and thus the number of mistakes) will be $\Theta(n^k)$ where $k = \max_{j \in [m]} k_j$. Based on our previous observation about the complexity of finding a minimal representation (in terms of items) consistent with the perceived conflicts, even when there is only one copy of each item, we suspect this problem may be inherent. We leave the question open of whether one can predict the winning sets of single-minded buyers with a mistake bound and running time which is polynomial in m, n and k (or even independent of k, for the mistake bound).

5. VARIABLE PRICES: ADDITIVE AND UNIT-DEMAND

The previous sections can be thought of simulating a simple mechanism: according to the mechanism, buyers have priorities, and in order of priority, buyers will pick her favorite bundle available. For single-minded buyers, the priorities could be thought of as a sorting of buyers by their bid, or some other pecking order. A buyer's preferences could be thought of as an ordering according to value, or quasilinear utility according to some fixed prices. We now consider a twist on our original setup: what if, rather than

the prices being fixed, each round was fed a price vector p^t along with the subset S^t, assuming that buyers would now take a bundle to maximize their quasilinear utility with respect to these varying prices? Assume, for simplicity, that $p^t \in \{0, 1, \ldots, V\}^m$. Simply running our previous mistake bound algorithm for unit-demand, additive, or single-minded buyers is tantalizingly simple. However, as buyer's preferences over bundles changes with prices (and thus rounds), this approach fails miserably.

The algorithm which solves the fixed price problem for unit-demand buyers can be thought of in a slightly different way, which will be useful for solving the problems with variable prices. An equivalent solution to the problem is to start with a permutation data structure P to learn the ordering over buyers, and for each buyer i, a permutation data structure P_i to learn their preference ordering over items. Whenever a mistake is made, the algorithm blames the subroutine which is learning some buyer's preferences (namely, the earliest buyer for which we made a mistake). If this causes their subroutine to become infeasible, it must be the case that the buyer needs to be demoted in the larger ordering, so we demote the buyer and restart their subroutine. Then, the total mistake bound for the algorithm will be $n^2\mathcal{MB}$, where $\mathcal{MB}$ is the mistake bound for the subroutines (because each buyer can be demoted at most n times, and there are at most $\mathcal{MB}$ mistakes for a buyer at each position). This intuition (running a global algorithm with subroutines for each buyer's preferences) is our starting point for constructing mistake-bound learning algorithms for variable prices. We begin by designing an algorithm for additive buyers, which gives some intuition for the case of unit-demand. We assume, for simplicity, there are no ties for a buyer's most-preferred bundle at any set of prices. All results can be extended to allow ties assuming buyers break ties consistently.

Additive. Suppose in each round t, our input is a subset of buyers present S^t and a price vector p^t. The algorithm $\mathcal{A}$ we are trying to simulate is composed of two parts, $\succ$, a priority over buyers, and $v_i(j)$ for each buyer $i \in B$, and $j \in I$, corresponding to the value buyer i has for item j. On a given subset and price vector pair, $\mathcal{A}$ will offer all items to buyer $i \in S^t$ who is first in $\succ$, who will take all items such that $v_i(j) > p^t(j)$. Then, the remaining items are offered to the remaining buyers $i' \in S^t$, in order of $\succ$, who will do the same.

Our algorithm will predict an allocation of item $\widehat{X}_1^t, \ldots, \widehat{X}_n^t$. If the allocation is wrong, the correct allocation $X_1^t, \ldots, X_n^t$ is shown to our algorithm. We wish, for arbitrary price vectors and subsets, to minimize the total number of errors made.

The argument of correctness for Algorithm AdditiveVariable (below) is somewhat more complex than in the previous sections. As before, we need to show that the algorithm never tells P to push down a buyer when it should not. The condition for this is slightly more complicated, however. That is, the infeasibility of the binary search for $v_i(j)$ is proof that this buyer cannot be above all of the buyers below him according to P. It is important that the first error according to $\texttt{Permutation}(P)$ (and *only* this error) is the one used to update the model: this avoids an earlier error in the ordering P_t (or an error in an earlier binary search $v_i(j)$) placing incorrect constraints on lower buyers. To this end, let $\texttt{FirstMistake}(X, X', P)$ denote the first i, according to P, for which $X_i \neq X'_i$. Let V be the maximum valuation any buyer has for any item.

THM 5.1. *Algorithm AdditiveVariable is an $O(\log(V)mn^2)$ mistake-bound learning algorithm for the problem of predicting subsets of quasilinear additive buyer's purchases according to $\succ$ priority with prices p^t.*

Lemma 5.1 is the main component of Theorem 5.1; it takes the place of the invariants used in the proof of Theorem 6.2, implying that no agent is demoted unless we are sure

Algorithm AdditiveVariable: MB algorithm predicting $X_1^t, \dots X_n^t$; additive buyers under order $\succ$

$P = \texttt{InitPerm}(n)$;
Let $\overline{v}_i(j) = V$; $\underline{v}_i(j) = 0$;
For $t = 1$ **to** T**:**
  Receive S^t, p^t
  Let $\widehat{X}_i^t = \emptyset$; $I' = [m]$
  Let $\widehat{v}_i^t(j) = \frac{\overline{v}_i(j) + \underline{v}_i(j)}{2}$
  for $b = 1$ *to* $|S^t|$ **do**
    // bth-ranked buyer of S^t w.r.t P
    Let $i = \texttt{Buyer}(b, S^t, P)$
    Let $X_i^t = \{j \in [m] | \widehat{v}_i(j) > p^t(j)\}$
    Let $I' = I' \setminus X_i^t$
  Predict $\widehat{X}_1^t, \dots, \widehat{X}_n^t$
  Learn $X_1^t, \dots, X_n^t$
  if $X^t \neq \widehat{X}^t$ **then**
    // index of first mistake according to P
    Let $i^t = \texttt{FirstMistake}(X^t, \widehat{X}^t, P)$;
    if $\exists j \in X_{i^t}^t \Delta \widehat{X}_{i^t}^t$ *such that* $\overline{v}_{i^t}(j) = \underline{v}_{i^t}(j)$ **then**
      // $\exists$ item whose binary search invalid
      $\texttt{Demote}(i^t, P)$, Let $\underline{v}_{i^t}(j) = 0$ and $\overline{v}_{i^t}(j) = V$;
    else
      // Update the binary searches.
      for *each* $j \in \widehat{X}_{i^t}^t \setminus X_{i^t}^t$ **do**
        Set $\overline{v}_{i^t}(j) = \widehat{v}_{i^t}(j)$;
      for *each* $j \in X_{i^t}^t \setminus \widehat{X}_{i^t}^t$ **do**
        Set $\underline{v}_{i^t}(j) = \widehat{v}_{i^t}(j)$;

her current position is too high, and that we always make progress when a mistake is made.

LEMMA 5.1. *Let buyer $i^t \in B$ be the first mistake in round t. Then, there exists some item $j \in I$ for which one of these statements holds:*

(1) $j \notin \widehat{X}_{i^t}^t$, but $j \in X_{i^t}^t$, and $\widehat{v}_{i^t}(j) < v_{i^t}(j)$
(2) $j \in \widehat{X}_t^{i^t}$, but $j \notin X_{i^t}^t$, and $\widehat{v}_{i^t}(j) > v_{i^t}(j)$
(3) $j \in \widehat{X}_{i^t}^t$, but $j \notin X_{i^t}^t$, and there is some i' such that $\texttt{Loc}(P^t, i^t) < \texttt{Loc}(P^t, i')$ but $\texttt{Loc}(\succ, i^t) > \texttt{Loc}(\succ, i')$.

We relegate the proofs of Lemma 5.1 and Theorem 5.1 to Section A.2.

Unit-Demand. The case of unit-demand buyers is similar to that of additive buyers, though the buyers will no longer have *separable* preferences over items: instead, out of a set of available items T at prices p^t, buyer i will buy $j = \text{argmax}_{j \in J}[v_i(j) - p^t(j)]$ to maximize his quasilinear utility (assume that there is some consistent tie-breaking in the event that several items are equally good). So, rather than using binary search for each item separately, for each buyer, we will run a mistake bound ellipsoid algorithm; whenever a constraint is added, it will be of the form $v_i(j) - p^t(j) > v_i(j') - p^t(j')$, where the $v_i(j)$s are variables and the $p^t(j)$s are constants coming from the online price vectors.

THM 5.2. *Algorithm UnitVariable is an $O(n^2 \mathcal{MB})$-mistake bound learner for unit-demand buyers with respect to some order $\succ$, where $\mathcal{MB}$ is the online mistake bound guarantee of the online classification algorithm.*

The main theorem of this section follows from a similar analysis to that of additive buyers in the previous section, with a twist stemming from the fact that we use the Ellipsoid algorithm as the mistake-bound subroutine (with each mistake serving as its separation oracle), rather than binary search for each item separately. This is similar

Algorithm UnitVariable: MB algorithm predicting $X_1^t, \ldots X_n^t$; unit-demand buyers, order $\succ$

$P = \texttt{InitPerm}(n)$
Let $\widehat{R}_i$ be an instance of ellipsoid alg. w. unknowns $v_i(j)$ for all $i \in B, j \in I$
For $t = 1$ **to** T**:**
  Receive S^t, p^t
  Let $\widehat{X}_i^t = \emptyset$ and $I' = [m]$
  for $b = 1$ *to* $|S^t|$ **do**
    Let $i = \texttt{Buyer}(b, S^t, P)$
    Let $\widehat{v}_i \in \mathbb{R}^m$ be the center estimated by $\widehat{R}_i$
    `// prediction for` i `w.r.t. prices, est. values, rem. items`
    Let $\widehat{j}_i = \text{argmax}_{j \in I'} \widehat{v}_i(j) - p^t(j)$
    if $\widehat{v}_i(\widehat{j}_i) - p^t(\widehat{j}_i) > 0$ **then**
      Let $X_i^t = \{\widehat{j}_i\}$
    else
      Let $X_i^t = \emptyset$
    Let $I' = I' \setminus X_i^t$
  Predict $\widehat{X}_1^t, \ldots, \widehat{X}_n^t$
  Learn $X_1^t, \ldots, X_n^t$
  if $X \neq \widehat{X}$ **then**
    Let $i^t = \texttt{FirstMistake}(X^t, \widehat{X}^t, P)$
    if $X_{i^t}^t = \emptyset$ **then**
      Send constraint $v_{i^t}(\widehat{j}_{i^t}) < p^t(j_{i^t})$ to $\widehat{R}_{i^t}$
    else
      Let $\{j_{i^t}\} = X_{i^t}^t$; // The item i^t actually won
      if $\widehat{X}_{i^t}^t = \emptyset$ **then**
        Send constraint $v_{i^t}(j_{i^t}) > p^t(j_{i^t})$ to $\widehat{R}_{i^t}$
      else
        Let $\widehat{X}_{i^t}^t = \{\widehat{j}_{i^t}\}$
        Send constraint $v_{i^t}(j_{i^t}) - v_{i^t}(\widehat{j}_{i^t}) > p^t(j_{i^t}) - p^t(\widehat{j}_{i^t})$ to $\widehat{R}_{i^t}$
      if $\widehat{R}_{i^t}$ *is infeasible* **then**
        $\texttt{Demote}(i^t, P)$ and restart $\widehat{R}_{i^t}$

to the use of the Ellipsoid algorithm by Maass and Turán [1990] for learning a linear separator. The proof of Theorem 5.2 can be found in Section A.2.

6. LOWER BOUNDS AND HARDNESS

Given the results in the previous section, it is natural to ask what mistake bounds cannot be achieved for these problems. This section addresses this question: we show information-theoretic lower bounds on the number of mistakes for several of these problems. For the case of single-minded bidders, our lower bound is matching the mistake bound we prove for an efficient algorithm. For unit-demand bidders with fixed prices, our lower bound is within a factor of n of the upper bound we have shown in the previous section using an efficient algorithm. We mention a computationally inefficient algorithm which achieves the information-theoretic optimal mistake bound , and leave it as an open question whether a polynomial-time algorithm can achieve the same mistake bound. Finally, we show cryptographic hardness of a related problem. Suppose bidders are unit-demand, and for each bidder i has a *set* of satisfying items, and our it is our goal to predict the set of agents who received one of their satisfying items. When our algorithm makes a mistake, it is presented with the set of satisfied agents, rather than the allocation. We show, by the reductions of Kearns and Valiant [1994], this problem is at least as hard as several encryption problems which are generally accepted as computationally hard problems, even in the average-case. We now state our information-theoretic lower bound for predicting the winning bidders for single-minded bidders.

THM 6.1. *Suppose bidders are single-minded and we predict and observe only which subset $W^t \subseteq S^t$ wins in round t. Then, when $m \geq n^2$, any online algorithm can be forced to make $\Omega(n^2)$ mistakes.*

PROOF. Suppose there are $\frac{n(n-1)}{2}$ items (one for each pair of buyers). In each round t, the adversary presents the algorithm with S^t containing a pair of buyers that has never been presented before. The algorithm needs to predict whether one or both of the buyers will be satisfied (guessing whether both buyers are both interested in their "shared" item or not). Regardless of the algorithm's choice, the adversary will say that was a mistake: this yields a consistent set of conflicts and will force the algorithm to make $\Omega(n^2)$ mistakes. □

On the other hand, there may be room for improving our unit-demand results. As a warm-up, we first present an upper and lower bound for the single item case (where we *do* have an efficient algorithm for the matching the upper bound).

THM 6.2. *The problem of learning the allocation made by a priority-ordered mechanism with fixed prices and a single item has a mistake bound $M = \Theta(n \log(n))$, and there is an algorithm with this mistake bound that runs in polynomial time.*

PROOF OF THEOREM 6.2. For the lower bound, an adversary can present subsets of size 2 and essentially simulate merge-sort. To start, for $i = 1, \ldots, n/2$, the adversary presents subset $\{2i-1, 2i\}$, and tells the algorithm it has made a mistake (regardless of its prediction), causing $n/2$ mistakes. In general, given n/L sorted lists of size L, the adversary pairs the lists together and then for each pair runs through the merging process (presenting the subset consisting of the top element in each list, telling the algorithm it has made a mistake whatever its prediction is, and popping off the true largest element). This maintains consistency with an overall ordering and creates at least L mistakes per pair, or again $n/2$ mistakes total for the round. There are $\log(n)$ rounds, leading to an overall lower bound of $\Omega(n \log n)$.

We can construct a computationally efficiently algorithm which matches this information-theoretic lower bound using two ideas. First, each mistake gives us a new pair of agents (i, j) for which we learn $i \succ j$ but $\texttt{Loc}(i, P^t) < \texttt{Loc}(j, P^t)$ (the true winner i has higher priority than every other $i' \in S^t$, and in particular, the estimated winner j). Second, Karzanov and Khachiyan [1991] given an efficient sampling algorithm which samples uniformly a consistent linear extension of a partial order.

Then, consider the following prediction algorithm. Consider a new subset S^t. Take a single sample $\succ_?$ using the algorithm of Karzanov and Khachiyan [1991], and predict the winner is $j^t = \texttt{Buyer}(1, S^t, \succ_?)$. If a mistake is made, and i^t is the winner, add the set of constraints $i^t \succ j$ for all $j \in S^t$ to the partial order. We claim that each constraint added to the partial order over the life of the algorithm is correct (they are added because a mistake is proof of the constraint). Second, when a mistake is made, the number of consistent linear extensions shrinks (multiplicatively) by at least $\frac{1}{4}$ in expectation. This fact follows from the fact that if there is some k^t whose probability of winning at time t is at least $\frac{1}{2}$ (where this probability is taken over the set of consistent linear extensions), there is probability at least $\frac{1}{2}$ of our algorithm predicting k^t. If k^t is incorrect, then all permutations where k^t is first amongst S^t are inconsistent after adding the new constraints, cutting the number of consistent linear extensions in half. Another winner is predicted with probability at most $\frac{1}{2}$, and the set of linear extensions only shrinks. Thus, by an analysis similar to the halving algorithm, after $\Theta(n \log(n))$ mistakes, there is only one consistent linear extension, and it is $\succ$. □

The case of unit-demand buyers also has matching information-theoretic bounds.

THM 6.3. *The problem of learning the allocation made by an priority-ordered mechanism with fixed prices with over unit-demand buyers has a mistake bound of* $\Theta(mn\log(m))$ *(assuming* $m = \Omega(\log(n))$*).*

PROOF OF THEOREM 6.3. The lower bound for this problem is similar to the previous argument. The generalization uses n buyers, the first m of which are "dummy" buyers and have favorite items $a_1, \ldots, a_m$. We can use these first buyers to control which items are available for the true buyers. Then, each example S^t will contain $m-2$ "dummy" buyers (who take all but just 2 items a_f, a_g) and one true buyer i. Then, the algorithm needs to decide which of a_f or a_g the true buyer will select. This will be repeated for each pair of items and each non-dummy buyer. Thus, the algorithm is solving $n-m$ separate instances of sorting m items (for each buyer), and so the problem has a lower bound of $\Omega(mn\log(m))$ mistakes. □

Without computational constraints, we can construct an algorithm with a matching mistake bound for unit-demand bidders. The algorithm will maintain a list of consistent permutations over buyers (and, for each of those permutations over buyers, the consistent permutations for each buyer over items). Given a new subset S^t, the algorithm predicts the most likely allocation (uniformly weighted). Since there are $n!*(m!)^n$ many initial hypotheses (an ordering over buyers and, for each buyer, an ordering over items), the halving algorithm will make $O(n\log(n)+nm\log(m))$ mistakes. It is not clear how to make this algorithm computationally efficient without increasing the mistake bound: unlike the single-item case, we have no clear culprit to our mistake. In the single-item case, we can add another constraint to our partial order, generating a refined partial order. In the unit-demand case, a mistake could be made either because the understanding of some individual's preferences are wrong, or because our ordering over buyers was wrong.

Unit-demand bidders with acceptable items is hard. Finally, we show hardness for a variant of the single-minded model (where we only observe which agents received their demanded sets, not the allocation itself). Suppose, rather than agent i being single-minded with demand set D_i, agent i is unit-demand, with preference order $\succ_i$ and an *acceptable set* D_i. If agent i is offered a set of items I, they will pick the item $j \in D_i \cap I$ (if the intersection is nonempty) which is best according to $\succ_i$. If agent i receives any item $j \in D_i$, we will say agent i is satisfied. Our goal is to predict the satisfied set W^t from S^t, the set of agents who arrive at time t. If a mistake is made, the algorithm is presented with the correct satisfied set (rather than the actual allocation). We call this problem the **unit-demand satisfaction prediction problem**.

THM 6.4. *Weakly-learning general boolean formulae of size* n *is polynomial-time reducible to the unit-demand satisfaction problem.*

We sketch a proof of Theorem 6.4 (a formal proof is in the full version of this paper).

PROOF SKETCH. One can encode a boolean formula (in its binary tree representation with and/or gates as internal nodes, and the leaves are inputs to the formula, the root's value is the value of the formula evaluated on its leaves) into an instance of the problem of predicting the set of satisfied unit-demand buyers. The intuitive correspondence is that each node will correspond to an item: that node will evaluate to true if and only if the corresponding item isn't chosen by some agent.

Each leaf node in a boolean formula will have a corresponding agent and item (that agent wants exactly that item). Each AND gate will introduce four bidders and four items. Two of the buyers each prefers one of two "input" items to the AND gate, then one of the non-input items associated with the gate. The other two buyers are each satisfied by only one of the non-input items corresponding to the gate; the preference

ordering gives higher priority to the first two buyers. These latter two buyers will be satisfied if and only if both inputs to the AND gate were true. Each OR gate introduces two additional bidders and three items; the first agent will choose either of the input items, then the third non-input item; the other bidder is satisfied only by the non-input item. If at least one of the input items is available, both agents will be satisfied. □

Thus, by the reduction of Kearns and Valiant [1994], there is a polynomial $p(n)$ such that inverting the RSA encryption function, recognizing quadratic residues, and factoring Blum integers are probabilistically polynomial-time reducible to the unit-demand satisfaction problem.

7. DISCUSSION

In this paper we present algorithms that from observations of opaque transactions (observing just who wins and who doesn't in the case of single-minded buyers, or observing the allocations produced in the case of additive or unit-demand buyers) can reconstruct both the preferences of the buyers and the mechanism used by the seller sufficiently well to predict the outcomes of new transactions. We focus on priority-based *ordered priority mechanisms* on the side of the seller, and commonly-studied classes of valuation functions for the buyers. It would be interesting to consider this problem in the context of other mechanisms and other observation models as well. Note that for mechanisms such as VCG (producing a social-welfare-maximizing allocation) certain complications arise: for instance even in the case that all buyer valuations are known, finding the allocation produced can be NP-complete if buyers are single-minded. Therefore, if one wishes to solve the prediction problem with efficient algorithms, it is necessary to consider settings where the problem of computing the allocation with full knowledge of the mechanism and preferences is polynomial time.

REFERENCES

AMIN, K., CUMMINGS, R., DWORKIN, L., KEARNS, M., AND ROTH, A. 2014. Online learning and profit maximization from revealed preferences. *arXiv*.

BALCAN, M.-F., DANIELY, A., MEHTA, R., URNER, R., AND VAZIRANI, V. V. 2014. Learning economic parameters from revealed preferences. *arXiv abs/1407.7937*.

BEIGMAN, E. AND VOHRA, R. 2006. Learning from revealed preference. In *Proceedings of the 7th ACM Conference on Electronic Commerce*. ACM, 36–42.

HELMBOLD, D., SLOAN, R., AND WARMUTH, M. K. 1990. Learning nested differences of intersection closed concept classes. *Machine Learning 5,* 2, 165–196. Special Issue on Computational Learning Theory; first appeared in 2nd COLT conference (1989).

KARZANOV, A. AND KHACHIYAN, L. 1991. On the conductance of order markov chains. *Order 8,* 1, 7–15.

KEARNS, M. AND VALIANT, L. 1994. Cryptographic limitations on learning boolean formulae and finite automata. *Journal of the ACM (JACM) 41,* 1, 67–95.

MAASS, W. AND TURÁN, G. 1990. *How fast can a threshold gate learn?* International Computer Science Institute.

SAMUELSON, P. A. 1938. A note on the pure theory of consumer's behaviour. *Economica*, 61–71.

VARIAN, H. R. 2006. Revealed preference. *Samuelsonian economics and the twenty-first century*, 99–115.

ZADIMOGHADDAM, M. AND ROTH, A. 2012. Efficiently learning from revealed preference. In *Internet and Network Economics*. Springer, 114–127.

A. MISSING PROOFS

A.1. Improved algorithm for unit-demand bidders

PROOF OF THEOREM 4.2. We describe how to efficiently implement an approximate halving algorithm for learning permutations consistent with a partial order in Section 6. If i^t is the first mistake, either the estimate of i^t's preferences are incorrect (in which case $j_{i^t} \succ_{i^t} \hat{j}_{i^t}$, and we add

Algorithm UnitDemandPrime: Predicts allocation of order-based allocation rule for unit-demand players

$P = \texttt{InitPerm}(n)$
for $i = 1$ *to* m **do**
  Let $\hat{\succ}_i$ be an instantiation of the apx. halving alg. (Section 6) for learning $\succ_i$ over $[m]$
For $t = 1$ **to** T**:**
  Let $I = [m]$, receive S^t
  for $b = 1$ *to* $|S^t|$ **do**
    Let $i = \texttt{Buyer}(b, S^t, P)$
    `// the predicted choice of item by` i
    Let $\hat{j}_i = \texttt{Buyer}(1, I, \hat{\succ}_i)$
    Let $\hat{X}_i^t = \{\hat{j}_i\}$ and $I = I \setminus \{\hat{j}_i\}$
Predict $\hat{X}^t$
Learn X^t
if $X^t \neq \hat{X}^t$ **then**
  `// ind of first error w.r.t.` P
  Let $i^t = \texttt{FirstMistake}(X^t, \widehat{X}^t, P)$
  Let $\{j_{i^t}\} = X_{i^t}^t$
  Give the constraint $j_{i^t} \succ_{i^t} \hat{j}_{i^t}$ to $\hat{\succ}_{i^t}$
  if $\hat{\succ}_{i^t}$ *is infeasible* **then**
    $\texttt{Demote}(i^t, P)$
    Reset $\hat{\succ}_{i^t}$

this constraint), or i^t needs to be demoted. Once $\hat{\succ}_{i^t}$ becomes infeasible, all constraints added were valid, so demoting i^t is valid. □

A.2. Improved Prices

PROOF OF LEMMA 5.1. A mistake implies there is some item j such that $j \notin \widehat{X}_{i^t}^t$ but $j \in X_{i^t}^t$, or $j \in \widehat{X}_{i^t}^t$ but $j \notin X_{i^t}^t$. Consider the first case. Our algorithm did not give item j to buyer i^t: moreover, it didn't give item j to some buyer i' with higher priority ($\texttt{Loc}(P^t, i') < \texttt{Loc}(P^t, i^t)$), since buyer i^t was the *first* mistake, implying our estimate of buyer i^t's value for item j was too low, which falls into case 1. If, on the other hand, our algorithm allocated item j to buyer i^t, but buyer i^t was not awarded item j, then either buyer i^t's value for item j is less than the price (and our estimate $\widehat{v}_{i^t}(j)$ was too high, implying case 2), or some buyer i' with higher priority w.r.t $\succ$ took item j before buyer i^t. In this case, since buyer i^t was the first mistake, this implies $\texttt{Loc}(P^t, i^t) < \texttt{Loc}(P^t, i')$ but $\texttt{Loc}(\succ, i^t) > \texttt{Loc}(\succ, i')$, implying case 3. □

LEMMA A.1. *Using the ellipsoid algorithm to learn the collection $\{v_i(j)\}_j$ has a mistake bound of $O(m^2(K + \log m))$ so long as each mistake returns a constraint such that its current hypothesis $\hat{v}_i$ is no longer feasible, where K is the maximum precision of the v_is.*

PROOF. We will have m variables corresponding to the valuations of buyer i to each of the m items. The Ellipsoid algorithm maintains an ellipsoid that contains the feasible region (the possible m-tuples of valuations consistent with observations so far) and proposes as its current hypothesis the center of that ellipsoid.

We use this center as a proposed valuation for buyer i until we make an error involving her. Once we make an error we identify a violated linear constraint, and we return it to the Ellipsoid algorithm, which then updates its ellipsoid and hypothesis.

In each iteration (mistake of the algorithm) the volume shrinks multiplicatively by a fraction of $1 - \frac{1}{m}$. The initial volume is at most $2^{O(m(K+\log m))}$. The final volume, assuming that there is a consistent valuation, is at least $2^{-O(m(K+\log m))}$. This implies that after at most $O(m^2(K+\log m))$ errors we reach a volume which is too small, and therefore there is no feasible valuation. □

We now state the analog to Lemma 5.1 for the unit-demand case.

LEMMA A.2. *Suppose the unit-demand algorithm makes a mistake at time t. Let i^t be the first mistake (according to $\widehat{\succ}^t$). Let $\widehat{j}_{i^t}$ be the item we predicted i^t to win (if any) and j_{i^t} the item i^t won (if any). Then one of these holds:*

(1) $v_{i^t}(\widehat{j}_{i^t}) - p^t(\widehat{j}_{i^t}) < 0 < \widehat{v}_{i^t}(\widehat{j}_{i^t}) - p^t(\widehat{j}_{i^t})$*, or* $v_{i^t}(\widehat{j}_{i^t}) < \widehat{v}_{i^t}(\widehat{j}_{i^t})$

(2) $v_{i^t}(\widehat{j}_{i^t}) - p^t(\widehat{j}_{i^t}) < v_{i^t}(j_{i^t}) - p^t(j_{i^t})$
(3) $v_{i^t}(j_{i^t}) > \widehat{v}_{i^t}(j_{i^t})$
(4) $\widehat{j}_{i^t}$ *was not available (*$\exists i'$ *s.t.* $\texttt{Loc}(P^t, i^t) < \texttt{Loc}(P^t, i')$ *but* $\texttt{Loc}(\succ, i^t) > \texttt{Loc}(\succ, i')$*).*

PROOF. Consider a mistake on buyer i^t. Either it is the case that (a) i^t bought nothing and we predicted she bought something, (b) she bought something and we predicted nothing, or (c) we predicted the wrong item.

(a) occurs only when $\widehat{j}$ was no longer available (i^t needs to be demoted, case 4) or $\widehat{j}$ was too expensive, $v_{i^t}(\widehat{j}) - p^t(\widehat{j}) < 0$ (case 1). (b) can only occur because our estimate of her value of an item was to small (case 3), since she is the first mistake it cannot be because we predicted that someone earlier took j. (c) occurs when either $\widehat{j}$ was not available (and i needs a demotion, case 4) or our estimate of utility was wrong (case 2). □

PROOF OF THEOREM 5.2. We claim the same two invariants are true of Algorithm UnitVariable as were true of Algorithm AdditiveVariable, since Lemma A.2 provides the analogous guarantees (namely, that when we make a mistake, we either get to add a constraint to some ellipsoid algorithm, or we get to demote some buyer). Thus, the algorithm is correct. Each instantiation of the ellipsoid algorithm makes at most $\mathcal{MB}$ mistakes before we demote and restart, and there are at most n^2 demotions total. Thus, at most $\mathcal{MB}n^2$ mistakes in total are made. □

Now, we prove Theorem 5.1.

PROOF. As in previous proofs, we show our algorithm maintains two invariants:

(1) For any buyer i, for each item j, at each time t, the binary search for $v_i(j)$ has been given only accurate upper and lower bounds for any permutation $\succ'$ such that i has not been demoted in $\succ'$ from $\widehat{\succ}^t$ (some other buyers may be demoted). So, if buyer i's precedence does not decrease, any $v_i(j)$s consistent with $\succ'$ and the observations are consistent with the binary searches and $\widehat{\succ}^t$.
(2) Any i^t demoted in P_t has some i' s.t. $\texttt{Loc}(P^t, i^t) < \texttt{Loc}(P^t, i')$ but $\texttt{Loc}(\succ, i') < \texttt{Loc}(\succ, i^t)$.

We begin with the first invariant. It is satisfied prior to any constraints being added to any buyer's binary searches. Now, suppose it is true until time t: all constraints are accurate w.r.t $\widehat{\succ}^t$ and any $\succ'$ such that i has not been demoted from $\widehat{\succ^t}$ to $\succ'$ but other buyers may have been demoted. If, at time t, i gets another constraint added to her binary searches, this implies either this constraint is correct or buyer i needs to be demoted, by Lemma 5.1. Thus, if buyer i is not demoted (as is the case for $\succ'$), this new constraint (and so all the constraints) in her binary searches are valid.

Now, we prove the second invariant. The invariant is true at the beginning of the algorithm prior to any buyer being pushed downwards. Now, consider some time t, and assume this invariant holds until time t. The only case of interest is when one of buyer i^t's binary searches is infeasible and she is demoted.

Due to invariant 1, we know that all the constraints buyer i^t has received until time t are valid at her position in $\widehat{\succ}^t$ (or any earlier position), since buyer i^t's binary searches are reset whenever buyer i^t is pushed down. Since she is the first mistake, by Lemma 5.1, it is either the case that the current constraint being added is true with respect to her $v_{i^t}(j)$s and her position (or an earlier position), or she must occur later in the ordering. Thus, all the constraints in her binary searches are correct with respect to her current position (or any earlier one) in the ordering. Since there are some $v_{i^t}(j)$s which are consistent with the observations, but not the set of constraints, it must be the case that buyer i^t occurs later in the ordering.

Now, we show the mistake bound. If a mistake is made, some buyer i^t either updates her binary searches or is demoted. At most $\log(V)$ binary search updates can occur for a given item and buyer before the binary search becomes infeasible and she is demoted. Thus, there can be at most $m\log(V)$ mistakes resulting in binary search updates for a buyer before she is demoted. By Lemma 3.1 and invariant 2, no buyer is pushed later in the ordering than she occurs in $\succ$; thus, there are at most n^2 mistakes resulting in demotions. Thus, in total, there are at most $n^2 m\log(V)$ many mistakes. □