Reducing Revenue to Welfare Maximization: Approximation Algorithms and Other Generalizations

Citation: Yang Cai, Constantinos Daskalakis, and S. Matthew Weinberg. 2013. Reducing revenue to welfare maximization: approximation algorithms and other generalizations. In Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '13). SIAM 578-595.

As Published: http://dl.acm.org/citation.cfm?id=2627859

Publisher: Association for Computing Machinery (ACM)

Persistent URL: http://hdl.handle.net/1721.1/99957

Version: Author's final manuscript: final author's manuscript post peer review, without publisher's formatting or copy editing

Reducing Revenue to Welfare Maximization: Approximation Algorithms and other Generalizations

Yang Cai*
EECS, MIT
ycai@csail.mit.edu

Constantinos Daskalakis†
EECS, MIT
costis@mit.edu

S. Matthew Weinberg‡
EECS, MIT
smw79@mit.edu

Abstract

It was recently shown in [12] that revenue optimization can be computationally efficiently reduced to welfare optimization in all multi-dimensional Bayesian auction problems with arbitrary (possibly combinatorial) feasibility constraints and independent additive bidders with arbitrary (possibly combinatorial) demand constraints. This reduction provides a poly-time solution to the optimal mechanism design problem in all auction settings where welfare optimization can be solved efficiently, but it is fragile to approximation and cannot provide solutions to settings where welfare maximization can only be tractably approximated. In this paper, we extend the reduction to accommodate approximation algorithms, providing an approximation preserving reduction from (truthful) revenue maximization to (not necessarily truthful) welfare maximization. The mechanisms output by our reduction choose allocations via black-box calls to welfare approximation on randomly selected inputs, thereby generalizing also our earlier structural results on optimal multi-dimensional mechanisms to approximately optimal mechanisms. Unlike [12], our results here are obtained through novel uses of the Ellipsoid algorithm and other optimization techniques over *non-convex regions*.

1 Introduction

The *optimal mechanism design* problem is a central problem in mathematical economics that has recently gained a lot of attention in computer science. The setting for this problem is simple: a seller has a limited supply of several items for sale and many buyers interested in these items. The problem is to design an auction for the buyers to play that will maximize the seller's revenue. While being able to solve this problem in the worst-case would be desirable, it is easy to see that there can't be any meaningful worst-case solution. Indeed, even in the simpler case of a single buyer and a single item, how would the seller sell the item to optimize her profit without any assumptions about the buyer who can, in principle, lie about his willingness to pay?

To cope with this impossibility, economists have taken a Bayesian approach, assuming that a prior distribution is known that determines the values of the buyers for each item (and each bundle of items), and aiming to optimize the seller's revenue in expectation over bidders sampled from this distribution. Under this assumption, Myerson solved the single-item version of the problem in a seminal paper on Bayesian mechanism design [27]. On the other hand, until very recently there had been very small progress on the "multi-dimensional" version of the problem, i.e. the setting where the seller has multiple *heterogeneous* items for sale [26]. This challenging, and more general problem is the focus of this paper.

We proceed to state the problem we study more precisely. First, as we move beyond single-item settings, one may want to consider settings with non-trivial *feasibility constraints* on which bidders may simultaneously receive which items. Here are some examples:

1. Maybe the items are baseball cards. Here, a feasible allocation should award each card to at most one bidder.
2. Maybe the items are houses. Here, a feasible allocation should award each house to at most one bidder, and to each bidder at most one house.
3. Maybe the items are links in a network, and all bidders have a desired source-destination pair. Here, a feasible allocation should award each link to at most one bidder, and to each bidder a simple path from their source to their destination (or nothing).

*Supported by NSF Award CCF-0953960 (CAREER) and CCF-1101491.

†Supported by a Sloan Foundation Fellowship, a Microsoft Research Faculty Fellowship and NSF Award CCF-0953960 (CAREER) and CCF-1101491.

‡Supported by a NSF Graduate Research Fellowship and NSF award CCF-1101491.

Following the notation of [12] we encapsulate the feasibility constraints that the auctioneer faces in a set system $\mathcal{F}$ on possible allocations. Namely, if the bidder set is $[m]$ and the item set is $[n]$ then an allocation of items to bidders can be represented by a subset $O \subseteq [m] \times [n]$. $\mathcal{F}$ is then a set-system $\mathcal{F} \subseteq 2^{[m]\times[n]}$, determining what allocations are allowed for the auctioneer to choose from.

With this notation in place, we formally state our problem, and note that virtually every recent result in the revenue-maximization literature [2, 6, 10, 11, 12, 13, 14, 15, 17, 21, 25] studies a special case of this problem (perhaps replacing Bayesian Incentive Compatibility with Incentive Compatibility). A detailed review of this literature is given in Section 1.2.

Revenue-Maximizing Multi-Dimensional Mechanism Design Problem (MDMDP): Given m distributions $\mathcal{D}_1, \ldots, \mathcal{D}_m$, supported on $\mathbb{R}^n$, over valuation vectors for n heterogenous items (possibly correlated across items), and feasibility constraints $\mathcal{F}$, output a Bayesian Incentive Compatible (BIC)[a] mechanism M whose allocation is in $\mathcal{F}$ with probability 1 and whose expected revenue is optimal relative to any other, possibly randomized, BIC mechanism when played by m additive bidders[b] whose valuation vectors are sampled from $\mathcal{D} = \times_i \mathcal{D}_i$.

[a] A mechanism is said to be BIC if it asks bidders to report their valuation to the mechanism and it is in each bidder's best interest to report truthfully, given that every other bidder does so as well. See Section 2 for a formal definition.

[b] A bidder is additive if their value for a bundle of items is the sum of their value for each item in the bundle.

It was recently shown that solving MDMDP under feasibility constraints $\mathcal{F}$ can be poly-time reduced to (the algorithmic problem of) maximizing social welfare under the same feasibility constraints $\mathcal{F}$, i.e. running the VCG allocation rule with constraints $\mathcal{F}$ [12]. This result implies that, for all $\mathcal{F}$'s such that maximizing social welfare can be solved efficiently, MDMDP can also be solved efficiently. On the other hand, the reduction of [12] is geometric and sensitive to having an exact algorithm for maximizing welfare, and this limits the span of mechanism design settings that can be tackled. In this work we extend this reduction, making it robust to approximation. Namely, we reduce the problem of approximating MDMDP to within a factor α to the problem of approximately optimizing social welfare to within the same factor α. Before stating our result formally, let us define the concept of a *virtual implementation* of an algorithm.

Definition 1. *Let A be a social welfare algorithm, i.e. an algorithm that takes as input a vector $(t_1, \ldots, t_m)$ of valuations (or types) of bidders and outputs an allocation $O \in \mathcal{F}$. A* virtual implementation of A *is defined by a collection of functions $f_1, \ldots, f_m$, such that $f_i : T_i \to \mathbb{R}^n$, where T_i is bidder i's type set. On input $(t_1, \ldots, t_m)$ the virtual implementation outputs $A(f_1(t_1), \ldots, f_m(t_m))$, i.e. instead of running A on the "real input" $(t_1, \ldots, t_m)$ it runs the algorithm on the "virtual input" $(f_1(t_1), \ldots, f_m(t_m))$ defined by the functions $f_1, \ldots, f_m$. The functions $f_1, \ldots, f_m$ are called* virtual transformations.

With this definition, we state our main result informally below, and formally as Theorem 6.1 of Section 6.

Informal Theorem 1. *Fix some arbitrary $\mathcal{F}$ and finite $T_1, \ldots, T_m$ and let $A : \times_i T_i \to \mathcal{F}$ be a (possibly randomized, not necessarily truthful) social welfare algorithm, whose output is in $\mathcal{F}$ with probability 1. Suppose that, for some $\alpha \leq 1$, A is an α-approximation algorithm to the social welfare optimization problem for $\mathcal{F}$, i.e. on all inputs $\vec{t}$ the allocation output by A has social welfare that is within a factor of α from the optimum for $\vec{t}$. Then for all $\mathcal{D}_1, \ldots, \mathcal{D}_m$ supported on $T_1, \ldots, T_m$ respectively, and all $\epsilon > 0$, given black-box access to A and without knowledge of $\mathcal{F}$, we can obtain an $(\alpha - \epsilon)$-approximation algorithm for MDMDP whose runtime is polynomial in the number of items, the number of bidder types (and* not *type profiles), and the runtime of A. Moreover, the allocation rule of the output mechanism is a distribution over virtual implementations of A.*

In addition to our main theorem, we provide in Section 6 extensions for distributions of infinite support and improved runtimes in certain cases, making use of techniques from [17]. We also show that our results still hold even in the presence of bidders with hard budget constraints. We remark that the functions defining a virtual implementation of a social welfare algorithm (Definition 1) may map a bidder type to a vector with negative coordinates. We require that the approximation guarantee of the given social welfare algorithm is still valid for inputs with negative coordinates. This is not a restriction for arbitrary downwards-closed $\mathcal{F}$'s, as any α-factor approximation algorithm that works for non-negative vectors can easily be (in a black-box way) converted to an α-factor approximation algorithm allowing arbitrary inputs.[1] But this is not necessarily true

[1] The following simple black-box transformation achieves this: first zero-out all negative coordinates in the input vectors; then call the approximation algorithm; in the allocation output by the algorithm un-allocate item j from bidder i if the corresponding coordinate is negative; this is still a feasible allocation as the setting is downwards-closed.

for non downwards-closed $\mathcal{F}$'s. If optimal social welfare cannot be tractably approximated (without concern for truthfulness) under arbitrary inputs, our result is not applicable.

Beyond Additive Settings: We note that the additivity assumption on the bidders' values for bundles of items is already general enough to model all settings that have been studied in the revenue-maximizing literature cited above, and already contains all unit-demand settings.

Beyond these settings that are already additive, we remark that we can easily extend our results to broader settings with minimal loss in computational efficiency. As an easy example, consider a single-minded combinatorial auction where bidder i is only interested in receiving some fixed subset S_i of items, or nothing, and has (private) value v_i for S_i. Instead of designing an auction for the original setting, we can design an auction for a single "meta-item" such that allocating the meta-item to bidder i means allocating subset S_i to bidder i. So bidder i has value v_i for the meta-item. The meta-item can be simultaneously allocated to several bidders. However, to faithfully represent the underlying setting, we define our feasibility constraints to enforce that we never simultaneously allocate the meta-item to bidders i and j if $S_i \cap S_j \neq \emptyset$. As there is now only one item, the bidders are trivially additive. So, the new setting faithfully represents the original setting, there is only 1 item, and the bidders are additive. So we can use our main theorem to solve this setting efficiently.

More generally, we can define the notion of *additive dimension* of an auction setting to be the minimum number of meta-items required so that the above kind of transformation can be applied to yield an equivalent setting whose bidders are additive. For example, the additive dimension of any setting with arbitrary feasibility constraints and additive bidders with arbitrary demand constraints is n. The additive dimension of a single-minded combinatorial auction setting is 1. The additive dimension of general (i.e. non single-minded) combinatorial auction settings, as well as all settings with risk-neutral bidders is at most 2^n (make a meta-item for each possible subset of items). In Appendix D we discuss the following observation and give examples of settings with low additive dimension, including settings where bidders have symmetric submodular valuations [4].

OBSERVATION 1. *In any setting with additive dimension d, Informal Theorem 1 holds after multiplying the runtime by a poly(d) factor, assuming that the transformation to the additive representation of the setting can be carried out computationally efficiently in the setting's specification.*

1.1 Approach and Techniques. Our main result, as well as those of [2, 11, 12], are enabled by an algorithmic characterization of *interim allocation rules* of auctions.[2] The benefit of working with the interim rule is, of course, the exponential (in the number of bidders) gain in description complexity that it provides compared to the ex post allocation rule, which specifies the behavior of the mechanism for every *vector* of bidders' types. On the other hand, checking whether a given interim rule is consistent with an auction is a non-trivial task. Indeed, even in single-item settings, where a necessary and sufficient condition for feasibility of interim rules had been known for a while [7, 8, 16], it was only recently that efficient algorithms were obtained [11, 2]. These approaches also generalized to serving many copies of an item with a matroid feasibility constraint on which bidders can be served an item simultaneously [2], but for more general feasibility constraints there seemed to be an obstacle in even defining necessary and sufficient conditions for feasibility [11], let alone checking them efficiently.

In view of this difficulty, it is quite surprising that a general approach for the problem was offered in [12]. The main realization was that, for arbitrary feasibility constraints, the set of feasible interim rules is a convex polytope, whose facets are accessible via black-box calls to an exact welfare optimizer for the same feasibility constraints. Such an algorithm can be turned into a separation oracle for the polytope and used to optimize over it with Ellipsoid. However, this approach requires use of an exact optimizer for welfare, making it computationally intractable in settings where optimal social welfare can only be tractably approximated.

Given only an approximation algorithm for optimizing social welfare, one cannot pin down the facets of the polytope of feasible interim rules exactly. Still, a natural approach could be to resign from the exact polytope of feasible interim rules, and let the approximation algorithm define a large enough sub-polytope. Namely, whenever the separation oracle of [12] uses the output of the social welfare optimizer to define a facet, make instead a call to the social welfare approximator and use its output to define the facet. Unfortunately, unless the approximation algorithm is a maximal-in-range algorithm, the separation oracle obtained does not necessarily define a polytope. In fact, the region is likely not even convex, taking away all the geometry that is

[2] The interim rule of an auction is the collection of marginal allocation probabilities $\pi_{ij}(t_i)$, defined for each item j, bidder i, and type t_i of that bidder, representing the probability that item j is allocated to bidder i when her type is t_i, and in expectation over the other bidders' types, the randomness in the mechanism, and the bidders' equilibrium behavior. See Section 2.

crucial for applying Ellipsoid.

Despite this, we show that ignoring the potential non-convexity, and running Ellipsoid with this "weird separation oracle" (called "weird" because it does not define a convex region) gives an approximation guarantee anyway, allowing us to find an approximately optimal interim rule with black-box access to the social welfare approximator. The next difficulty is that, after we find the approximately optimal interim rule, we still need to find an auction implementing it. In [12] this is done via a geometric algorithm that decomposes a point in the polytope of feasible interim rules into a convex combination of its corners. Now that we have no polytope to work with, we have no hope of completing this task. Instead, we show that for any point $\vec{\pi}$ deemed feasible by our weird separation oracle, the black-box calls made during the execution to the social welfare approximator contain enough information to decompose $\vec{\pi}$ into a convex combination of virtual implementations of the approximation algorithm (which are not necessarily extreme points, or even contained in the region defined by our weird separation oracle). After replacing the separation oracle of [12] with our weird separation oracle, and the decomposition algorithm with this new decomposition approach, we obtain the proof of our main theorem (Informal Theorem 1 above, and Theorem 6.1 in Section 6). Our approach is detailed in Sections 3, 4 and 5.

1.2 Related Work

1.2.1 Optimal Mechanism Design. In his seminal paper, Myerson solved the single-item case of the MDMDP [27]. Shortly after, the result was extended to all "single-dimensional settings," where the seller has multiple copies of the same item and some feasibility constraint $\mathcal{F}$ on which of the bidders can simultaneously receive a copy. The algorithmic consequence of these results is that, for all $\mathcal{F}$'s such that social welfare can be (not necessarily truthfully) efficiently optimized, the revenue-optimal auction can also be efficiently found, and run. On the other hand, before this work, there was no analogue of this for approximation algorithms, allowing a generic reduction from revenue approximation to (not necessarily truthful) social-welfare approximation.

On the multi-dimensional front, where there are multiple, heterogeneous items for sale, progress had been slower [26], and only recently computationally efficient constant factor approximations for special cases were obtained [14, 6, 15, 1, 25, 28]. These results cover settings where the bidders are unit-demand and the seller has matroid or matroid-intersection constraints on which bidders can simultaneously receive items, or the case of additive-capacitated bidders, i.e. settings that are special cases of the MDMDP framework.[3] More recently, computationally efficient optimal solutions were obtained for even more restricted cases of MDMDP [2, 11, 17], until a general, computationally efficient reduction from revenue to welfare optimization was given in [12]. This result offers the analog of Myerson's result for multi-dimensional settings. Nevertheless, the question still remained whether there is an approximation preserving reduction from revenue to (not necessarily truthful) welfare optimization. This reduction is precisely what this work provides, resulting in approximately optimal solutions to MDMDP for all settings where maximizing welfare is intractable, but approximately optimizing welfare (without concern for truthfulness) is tractable.

1.2.2 Black-Box Reductions in Mechanism Design. Our reduction from approximate revenue optimization to non-truthful welfare approximation is a black-box reduction. Such reductions have been a recurring theme in mechanism design literature but only for *welfare*, where approximation-preserving reductions from truthful welfare maximization to non-truthful welfare maximization have been provided [9, 3, 23, 18, 5, 22]. The techniques used here are orthogonal to the main techniques of these works. In the realm of black-box reductions in mechanism design, our work is best viewed as "catching up" the field of revenue maximization to welfare maximization, for the settings covered by the MDMDP framework.

1.2.3 Weird Separation Oracle, Approximation, and Revenue Optimization. Grötschel et al. [19] show that exactly optimizing any linear function over a bounded polytope P is equivalent to having a separation oracle for P. This is known as the equivalence of exact separation and optimization. Jansen extends this result to accommodate approximation [24]. He shows that given an approximation algorithm $\mathcal{A}$ such that for any direction $\vec{w}$, $\mathcal{A}$ returns an approximately extreme point $\mathcal{A}(\vec{w}) \in P$, where $\mathcal{A}(\vec{w})\cdot\vec{w} \geq \alpha\cdot\max_{\vec{v}\in P}\{\vec{w}\cdot\vec{v}\}$, one can construct a strong, approximate separation oracle which either asserts that a given point $\vec{x} \in P$ or outputs a hyperplane that separates $\vec{x}$ from αP (the polytope P shrunk by α). We show a similar but stronger result. Under the same conditions, our weird separation oracle either outputs a hyperplane separating $\vec{x}$ from a polytope P_1 that contains αP, or asserts that $\vec{x} \in P_2$, where

[3] In some of these results, bidders may also have budget constraints (and this does not directly fit in the MDMDP framework). Nevertheless, budgets can be easily incorporated to the framework without any loss, as was shown in [12].

P_2 is a polytope contained in P. A precise definition of P_1 and P_2 is given in Section 3.1. Moreover, for any point $\vec{x}$ that the weird separation oracle asserts is in P_2, we show how to decompose it into a convex combination of points of the form $\mathcal{A}(\vec{w})$. This is crucial for us, as our goal is not just to find an approximately optimal reduced form, but also to implement it. The technology of [24] is not enough to accomplish this, which motivates our stronger results.

But there is another, crucial reason that prevents using the results of [24], and for that matter [19] (for the case $\alpha = 1$), as a black box for our purposes. Given a computationally efficient, α-approximate social-welfare algorithm A for feasibility constraints $\mathcal{F}$, we are interested in obtaining a separation oracle for the polytope $P = F(\mathcal{F}, \mathcal{D})$ of feasible interim allocation rules of auctions that respect $\mathcal{F}$ when bidder types are drawn from distribution $\mathcal{D}$. To use [24] we need to use A to come up with an α-approximate linear optimization algorithm for P. But, in fact, we do not know how to find such an algorithm efficiently for general $\mathcal{F}$, due to the exponentiality of the support of $\mathcal{D}$ (which is a product distribution over $\mathcal{D}_1, \ldots, \mathcal{D}_m$). Indeed, given $\vec{w}$ we only know how to query A to obtain some $\pi^*(\vec{w})$ such that $\pi^*(\vec{w}) \cdot \vec{w} \geq \alpha \cdot \max_{\vec{\pi} \in P}\{\vec{w} \cdot \vec{\pi}\} - \epsilon$, for some small $\epsilon > 0$. This additive approximation error that enters the approximation guarantee of our linear optimization algorithm is not compatible with using the results of [24] or [19] as a black box, and requires us to provide our own separation to optimization reduction, together with additional optimization tools.

2 Preliminaries and notation

2.1 MDMDP. Here are preliminaries and notation regarding the MDMDP aspect of our results. We use the same notation as [12]. Denote the number of bidders by m, the number of items by n, and the type space of bidder i by T_i. To ease notation, we sometimes use B (C, D, etc.) to denote possible types of a bidder (i.e. elements of T_i), and use t_i for the random variable representing the instantiated type of bidder i. So when we write $\Pr[t_i = B]$, we mean the probability that bidder i's type is B. The elements of $\times_i T_i$ are called *type profiles*, and specify a type for every bidder. We assume type profiles are sampled from a known distribution $\mathcal{D}$ over $\times_i T_i$. We denote by $\mathcal{D}_i$ the marginal of this distribution on bidder i's type, and use $\mathcal{D}_{-i}(B)$ to denote the marginal of $\mathcal{D}$ over the types of all bidders except i, conditioned on $t_i = B$. If $\mathcal{D}$ is a product distribution, we will drop the parameter B and just write $\mathcal{D}_{-i}$. We refer the reader to Appendix C for a discussion on how an algorithm might access the distribution $\mathcal{D}$.

Let $[m] \times [n]$ denote the set of possible *assignments* (i.e. the element (i, j) denotes that bidder i was awarded item j). We call (distributions over) subsets of $[m] \times [n]$ (randomized) *allocations*, and functions mapping type profiles to (possibly randomized) allocations *allocation rules*. We call an allocation combined with a price charged to each bidder an *outcome*, and an allocation rule combined with a pricing rule a (direct revelation, or direct) *mechanism*. As discussed in Section 1, we may also have a set system $\mathcal{F}$ on $[m] \times [n]$ (that is, a subset of $2^{[m]\times[n]}$), encoding constraints on what allocations are feasible. $\mathcal{F}$ may be incorporating arbitrary demand constraints imposed by each bidder, and supply constraints imposed by the seller, and will be referred to as our *feasibility constraints*. In this case, we restrict all allocation rules to be supported on $\mathcal{F}$. We always assume that $\emptyset \in \mathcal{F}$, i.e. the auctioneer has the option to allocate no item.

The interim allocation rule, also called *reduced form* of an allocation rule, is a vector function $\pi(\cdot)$, specifying values $\pi_{ij}(B)$, for all items j, bidders i and types $B \in T_i$. $\pi_{ij}(B)$ is the probability that bidder i receives item j when truthfully reporting type B, where the probability is over the randomness of all other bidders' types (drawn from $\mathcal{D}_{-i}(B)$) and the internal randomness of the allocation rule, assuming that the other bidders report truthfully their types. Sometimes, we will want to think of the reduced form as a $n\sum_{i=1}^{m} |T_i|$-dimensional vector, and may write $\vec{\pi}$ to emphasize this view. To ease notation we will also denote by $T := n\sum_i |T_i|$.

Given a reduced form π, we will be interested in whether the form is "feasible", or can be "implemented." By this we mean designing a feasible allocation rule M (i.e. one that respects feasibility constraints $\mathcal{F}$ on every type profile with probability 1 over the randomness of the allocation rule) such that the probability that bidder i receives item j when truthfully reporting type B is exactly $\pi_{ij}(B)$, where the probability is computed with respect to the randomness in the allocation rule and the randomness in the types of the other bidders (drawn from $\mathcal{D}_{-i}(B)$), assuming that the other bidders report truthfully. While viewing reduced forms as vectors, we denote by $F(\mathcal{F}, \mathcal{D})$ the set of feasible reduced forms when the feasibility constraints are $\mathcal{F}$ and bidder types are sampled from $\mathcal{D}$.

A bidder is *additive* if her value for a bundle of items is the sum of her values for the items in that bundle. To specify the preferences of additive bidder i, we can provide a valuation vector $\vec{v}_i$, with the convention that v_{ij} represents her value for item j. Even in the presence of arbitrary (possibly combinatorial) demand constraints, the *value* of bidder i of type $\vec{v}_i$ for a randomized al-

location that respects the bidder's demand constraints with probability 1, and whose expected probability of allocating item j to the bidder is π_{ij}, is just the bidder's expected value, namely $\sum_j v_{ij} \cdot \pi_{ij}$. The *utility* of bidder i for the same allocation when paying price p_i is just $\sum_j v_{ij} \cdot \pi_{ij} - p_i$. Such bidders whose value for a distribution of allocations is their expected value for the sampled allocation are called *risk-neutral*. Bidders subtracting price from expected value are called *quasi-linear*.

Throughout the paper we denote by A a (possibly randomized, non-truthful) social welfare algorithm that achieves an α-fraction of the optimal welfare for feasibility constraints $\mathcal{F}$. We denote by $A(\{f_i\}_i)$ the virtual implementation of A with virtual transformations f_i (see Definition 1).

Some arguments will involve reasoning about the *bit complexity* of a rational number. We say that a rational number has bit complexity b if it can be written with a binary numerator and denominator that each have at most b bits. We also take the bit complexity of a rational vector to be the total number of bits required to describe its coordinates. Similarly, the bit complexity of an explicit distribution supported on rational numbers with rational probabilities is the total number of bits required to describe the points in the support of the distribution and the probabilities assigned to each point in the support. For our purposes the bidder distributions $\mathcal{D}_1, \ldots, \mathcal{D}_m$ are given explicitly, while $\mathcal{D} = \times_i \mathcal{D}_i$ is described implicitly as the product of $\mathcal{D}_1, \ldots, \mathcal{D}_m$.

Finally, for completeness, we define in Appendix B the standard notions of Bayesian Incentive Compatibility (BIC) and Individual Rationality (IR) of mechanisms.

2.2 Weird Separation Oracles. In our technical sections, we will make use of "running the ellipsoid algorithm with a weird separation oracle." A weird separation oracle is just an algorithm that, on input $\vec{x}$, either outputs "yes," or a hyperplane that $\vec{x}$ violates. We call it "weird" because the set of points that will it accepts is not necessarily convex, or even connected, so it is not a priori clear what it means to run the ellipsoid algorithm with a weird separation oracle. When we say "run the ellipsoid algorithm with a weird separation oracle" we mean:

1. Find a meaningful ellipsoid to start with (this will be obvious for all weird separation oracles we define, so we will not explicitly address this).
2. Query the weird separation oracle on the center of the current ellipsoid. If it is accepted, output it as a feasible point. Otherwise, update the ellipsoid using the violated hyperplane (in the same manner that the standard ellipsoid algorithm works).
3. Repeat step 2) for a pre-determined number of iterations N (N will be chosen appropriately for each weird separation oracle we define). If a feasible point is not found after N iterations, output "infeasible."

It is also important to note that we are *not* using the ellipsoid algorithm as a means to learning whether some non-convex set is empty. We are using properties of the ellipsoid algorithm with carefully chosen weird separation oracles to learn information, not necessarily related to a feasibility question.

3 The Weird Separation Oracle (WSO)

In this section, we take a detour from mechanism design, showing how to construct a weird separation oracle from an algorithm that approximately optimizes linear functions over a convex polytope. Specifically, let P be a bounded polytope containing the origin, and let $\mathcal{A}$ be any algorithm that takes as input a linear function f and outputs a point $\vec{x} \in P$ that approximately optimizes f (over P). We will define our weird separation oracle using black-box access to $\mathcal{A}$ and prove several useful properties that will be used in future sections. We begin by discussing three interesting convex regions related to P in Section 3.1. This discussion provides insight behind why we might expect WSO to behave reasonably. In addition, the polytopes discussed will appear in later proofs. In Section 3.2 we define WSO formally and prove several useful facts about executing the ellipsoid algorithm with WSO. For this section, we will not address running times, deferring this to Section 5. Our basic objects for this section are encapsulated in the following definition.

Definition 2. *P is a convex d-dimensional polytope contained in $[-1,1]^d$, $\alpha \leq 1$ is an absolute constant, and $\mathcal{A}$ is an approximation algorithm such that for any $\vec{w} \in \mathbb{R}^d$, $\mathcal{A}(\vec{w}) \in P$ and $\mathcal{A}(\vec{w}) \cdot \vec{w} \geq \alpha \cdot \max_{\vec{x} \in P}\{\vec{x} \cdot \vec{w}\}$. (Since $\vec{0} \in P$, this is always non-negative.)*

Notice that the restriction that $P \subseteq [-1,1]^d$ is without loss of generality as long as P is bounded, as in this section we deal with multiplicative approximations.

3.1 Three Convex Regions. Consider the following convex regions, where $Conv(S)$ denotes the convex hull of S.

- $P_0 = \{\vec{\pi} \mid \frac{\vec{\pi}}{\alpha} \in P\}$.
- $P_1 = \{\vec{\pi} \mid \vec{\pi} \cdot \vec{w} \leq \mathcal{A}(\vec{w}) \cdot \vec{w},\ \forall \vec{w} \in [-1,1]^d\}$.
- $P_2 = Conv(\{\mathcal{A}(\vec{w}),\ \forall \vec{w} \in [-1,1]^d\})$.

It is not hard to see that, if $\mathcal{A}$ always outputs the exact optimum (i.e. $\alpha = 1$), then all three regions are the same. It is this fact that enables the equivalence of separation and optimization [19]. It is not obvious, but perhaps not difficult to see also that if $\mathcal{A}$ is a maximal-in-range algorithm,[4] then $P_1 = P_2$. It turns out that in this case, WSO (as defined in Section 3.2) actually defines a polytope. We will not prove this as it is not relevant to our results, but it is worth observing where the complexity comes from. We conclude this section with a quick lemma about these regions, whose proof appears in Appendix E.1.

LEMMA 3.1. $P_0 \subseteq P_1 \subseteq P_2$.

3.2 WSO. Before defining WSO, let's state the properties we want it to have. First, for any challenge $\vec{\pi}$, WSO should either assert $\vec{\pi} \in P_2$ or output a hyperplane separating $\vec{\pi}$ from P_1. Second, for any $\vec{\pi}$ such that $WSO(\vec{\pi}) =$ "yes", we should be able to decompose $\vec{\pi}$ into a convex combination of points of the form $\mathcal{A}(\vec{w})$. Why do we want these properties? Our goal in later sections is to write a LP that will use WSO for $F(\mathcal{F}, \mathcal{D})$ to find a reduced form auction whose revenue is at least αOPT. Afterwards, we have to find an actual mechanism that implements this reduced form. So WSO needs to guarantee two things: First, running a revenue maximizing LP with WSO must terminate in a reduced form with good revenue. Second, we must be able to implement any reduced form that WSO deems feasible. Both claims will be proved in Section 4 using lemmas proved here. That using WSO achieves good revenue begins with Fact 3.1. That we can implement any reduced form deemed feasible by WSO begins with Lemma 3.2. We define WSO in Figure 1.

Let's now understand what exactly WSO is trying to do. What WSO really wants is to act as a separation oracle for P_2. As P_2 is a polytope, if $\vec{\pi} \notin P_2$, then there is some weight vector $\vec{w}$ such that $\vec{\pi} \cdot \vec{w} > \max_{\vec{x} \in P_2}\{\vec{x} \cdot \vec{w}\}$. WSO wants to find such a weight vector or prove that none exists (and therefore $\vec{\pi} \in P_2$). It is shown in [19] that if we replace $\mathcal{A}(\vec{w})$ with $\text{argmax}_{\vec{x} \in P_2}\{\vec{x} \cdot \vec{w}\}$ inside $\widehat{WSO}$, then WSO would be a separation oracle for P_2. Unfortunately, unless $\mathcal{A}$ is maximal-in-range, we cannot find $\text{argmax}_{\vec{x} \in P_2}\{\vec{x} \cdot \vec{w}\}$ with only black-box access to $\mathcal{A}$.[5] So WSO makes its best guess that $\mathcal{A}(\vec{w})$ is the maximizer it is looking for. Of course, this is not necessarily the case, and this is why the set of points accepted by WSO is not necessarily a convex region.

[4] Let S denote the set of vectors that are ever output by $\mathcal{A}$ on any input. Then $\mathcal{A}$ is maximal-in-range if, for all $\vec{w}$, $\mathcal{A}(\vec{w}) \in \text{argmax}_{\vec{x} \in S}\{\mathcal{A}(\vec{x}) \cdot \vec{w}\}$.

[5] If $\mathcal{A}$ is maximal-in-range, then this is exactly $\mathcal{A}(\vec{w})$.

$WSO(\vec{\pi}) =$

- "**Yes**" if the ellipsoid algorithm with N iterations[a] outputs "infeasible" on the following problem:

  variables: $\vec{w}, t$;

  constraints:

  - $\vec{w} \in [-1, 1]^d$;
  - $t - \vec{\pi} \cdot \vec{w} \leq -\delta$;[b]
  - $\widehat{WSO}(\vec{w}, t) =$
    - "yes" if $t \geq \mathcal{A}(\vec{w}) \cdot \vec{w}$;[c]
    - the violated hyperplane $t' \geq \mathcal{A}(\vec{w}) \cdot \vec{w}'$ otherwise.
- If a feasible point $(t^*, \vec{w}^*)$ is found, output the violated hyperplane $\vec{w}^* \cdot \vec{\pi}' \leq t^*$.

[a] The appropriate choice of N for our use of WSO is provided in Corollary 5.1 of Section 5. The only place that requires an appropriate choice of N is the proof of Lemma 3.2.

[b] The appropriate choice of δ for our use of WSO is provided in Lemma 5.1 of Section 5. The only place that requires an appropriate choice of δ is the proof of Lemma 3.2.

[c] Notice that the set $\{(\vec{w}, t) | \widehat{WSO}(\vec{w}, t) =$ "Yes"$\}$ is not necessarily convex or even connected.

Figure 1: A "weird" separation oracle.

Now, we need to prove some facts about WSO despite this. All proofs can be found in Appendix E.2.

FACT 3.1. *Consider an execution of the ellipsoid algorithm using WSO, possibly together with additional variables and constraints. Let Q be the polytope defined by the halfspaces output by WSO during its execution. Then during the entire execution, $P_1 \subseteq Q$.*

FACT 3.2. *If $\vec{\pi} \in P_1$, then $WSO(\vec{\pi}) =$ "yes."*

COROLLARY 3.1. *When WSO rejects $\vec{\pi}$, it acts as a valid separation oracle for P_1, or any polytope contained in P_1 (i.e. the hyerplane output truly separates $\vec{\pi}$ from P_1). In other words, the only difference between WSO and a valid separation oracle for P_1 is that WSO may accept points outside of P_1.*

LEMMA 3.2. *Let $WSO(\vec{\pi}) =$ "yes" and let S denote the set of weights $\vec{w}$ such that WSO queried $\widehat{WSO}(\vec{w}, t)$ for some t during its execution. Then $\vec{\pi} \in Conv(\{\mathcal{A}(\vec{w}) | \vec{w} \in S\})$.*

4 Approximately Maximizing Revenue using WSO

In this section, we show that running the revenue maximizing LP of [12] using the weird separation oracle of the

previous section obtains good revenue, and outputs a reduced form that can be implemented with only black-box access to a social welfare algorithm A.

In brush strokes, the approach of [12] is the following. They start by creating a proxy distribution $\mathcal{D}'$ that is a (correlated across bidders) uniform distribution over poly$(n, T, 1/\epsilon)$ type profiles. Roughly, $\mathcal{D}'$ is obtained by sampling the same number of profiles from $\mathcal{D}$, and forming the uniform distribution over them, and its advantage over $\mathcal{D}$ is that its support is polynomial. With $\mathcal{D}'$ at hand, it shown that the LP of Figure 2 in Appendix F outputs a reduced form whose revenue is at least OPT $- \epsilon$. This is proved by showing that the polytopes $F(\mathcal{F}, \mathcal{D})$ and $F(\mathcal{F}, \mathcal{D}')$ are "ϵ-close" in some meaningful way. To show how we adapt this approach to our setting, we need a definition.

Definition 3. *Let $\vec{w} \in \mathbb{R}^T$, and $\hat{\mathcal{D}}$ be a (possibly correlated) distribution over bidder type profiles. Define $f_i : T_i \to \mathbb{R}^n$ so that $f_{ij}(B) = \frac{w_{ij}(B)}{\Pr[t_i = B]}$. Then $A_{\hat{\mathcal{D}}}(\vec{w})$ denotes the allocation rule $A(\{f_i\}_i)$, $R^A_{\hat{\mathcal{D}}}(\vec{w})$ denotes the reduced form of $A_{\hat{\mathcal{D}}}(\vec{w})$, and $W^A_{\hat{\mathcal{D}}}(\vec{w}) := R^A_{\hat{\mathcal{D}}}(\vec{w}) \cdot \vec{w}$ is exactly the expected virtual welfare obtained by algorithm A under the virtual transformations $\{f_i\}_i$. For the purpose of the dot product, recall that we may view reduced forms as T-dimensional vectors (Section 2).*

Given this definition, and for the same $\mathcal{D}'$ used in [12], we let $P = F(\mathcal{F}, \mathcal{D}')$, and $\mathcal{A}(\vec{w})$ be the algorithm that on input $\vec{w} \in \mathbb{R}^T$ returns $R^A_{\mathcal{D}'}(\vec{w})$. Because taking a dot product with $\vec{w}$ is exactly computing expected virtual welfare (as in Definition 3), it is clear that $\mathcal{A}$ is an α-factor approximation algorithm for optimizing any linear function $\vec{w} \cdot \vec{x}$ over $\vec{x} \in P$. Using $\mathcal{A}$, we define P_0, P_1 and P_2 as in Section 3.

We continue to bound the revenue of the reduced form output by our LP of Figure 2. Denote by $Rev(F)$ the revenue obtained by the LP of Figure 2, and by $Rev(P_i)$ the revenue obtained by replacing P with P_i. The proof of Lemma 4.1, as well as all other results of this section are in Appendix F.

Lemma 4.1. $Rev(P_0) \geq \alpha Rev(F) \geq \alpha(OPT - \epsilon)$.

Now, denote by $Rev(WSO)$ the revenue obtained by replacing P with WSO in Figure 2. By "replace P with WSO," we mean run the optimization version of the ellipsoid algorithm that does a binary search on possible values for the objective function. On each subproblem (i.e. for a guess x of the revenue), run the ellipsoid algorithm using a new weird separation oracle WSO', which does the following. For challenge $(\vec{\pi}, \vec{p})$, first check if it satisfies the IR and BIC constraints in Figure 2 and the revenue constraint $\sum_{i=1}^m \sum_{\vec{v}_i \in T_i} \Pr[t_i = \vec{v}_i] \cdot p_i(\vec{v}_i) \geq x$, for the guessed value x of revenue. If not, output the hyperplane it violates. If yes, output $WSO(\vec{\pi})$. The ellipsoid algorithm will use **exactly the same parameters as if WSO was a separation oracle for P_0.** In particular, we can calculate the number of iterations and the precision that Ellipsoid would use if it truly had access to a separation oracle for P_0,[6] and use the same number of iterations here. Moreover, we use here the same criterion for deeming the feasible region lower-dimensional that the Ellipsoid with separation oracle for P_0 would use. Similarly, the bit complexity over values of x that the binary search will search over is taken to be the same as if binary search and the ellipsoid algorithm were used to solve the LP of Figure 2 with P_0 in place of $F(\mathcal{F}, \mathcal{D}')$.

We now want to use Lemma 4.1 to lower bound $Rev(WSO)$. This is almost a direct corollary of Fact 3.1. The only remaining step is understanding the ellipsoid algorithm.

Proposition 4.1. *If $x \leq Rev(P_0)$, then the ellipsoid algorithm using WSO' (with the same parameters as if WSO was a separation oracle for P_0) will always find a feasible point.*

Corollary 4.1. $Rev(WSO) \geq Rev(P_0)$.

Corollary 4.2. $Rev(WSO) \geq \alpha(OPT - \epsilon)$.

Finally, we need to argue that we can implement any reduced form output by the LP with WSO, as otherwise the reduced form is useless. This is a direct consequence of Lemma 3.2:

Corollary 4.3. *Let $\vec{\pi}^*$ denote the reduced form output by the LP of Figure 2 using WSO instead of $F(\mathcal{F}, \mathcal{D}')$, and let S be the set of weights $\vec{w}$ that are queried to $\widehat{WSO}$ during the execution. Then $\vec{\pi}^*$ can be implemented (for bidders sampled from $\mathcal{D}'$) as a distribution over virtual implementations of A using only virtual transformations corresponding to weights in S.*

At this point, we have shown that the reduced form $\vec{\pi}^*$ and pricing rule p^* computed by the LP of Figure 2 after replacing $F(\mathcal{F}, \mathcal{D}')$ with WSO achieves good revenue when bidders are sampled from $\mathcal{D}$, and define a BIC mechanism when bidders are sampled from $\mathcal{D}'$. We have also shown that we can implement $\vec{\pi}^*$ as a distribution over virtual implementations of A using only weights that were queried during the execution of the LP, *albeit for bidders are sampled from $\mathcal{D}'$.*

[6] These parameters were computed in [12] except for $F(\mathcal{F}, \mathcal{D}')$ rather than P_0. As the latter is just the former scaled by α it is easy to modify these parameters to accommodate α. This is addressed in Lemma 5.2 in Section 5.

The remaining step for correctness (we still have not addressed running time) is to show that, with high probability, the same distribution over virtual implementations of A implements some reduced form $\vec{\pi}'$ when the bidders are sampled from $\mathcal{D}$ that satisfies $|\vec{\pi}^* - \vec{\pi}'|_1 \leq \epsilon$. Once we show this, we will have proved that our distribution over virtual implementations of A and our pricing rule p^* define an ϵ-BIC, ϵ-IR mechanism when bidders are sampled from $\mathcal{D}$ with good revenue. We will argue this informally in Appendix F.1 and refer the reader to [12] for a formal proof of the same fact when using $F(\mathcal{F}, \mathcal{D}')$ rather than WSO in the LP of Figure 2 as the proof is nearly identical. In addition, we can give every bidder type an ϵ rebate in order to get an ϵ-BIC, IR mechanism for bidders sampled from $\mathcal{D}$ for an additional hit of $m\epsilon$ in revenue. (Recall that the runtime we are shooting for is polynomial in $1/\epsilon$, so ϵ can be made small enough to cancel the additional factor of m.) With this discussion, we have shown that our algorithm is correct: we have implemented some ϵ-BIC, IR mechanism $(\vec{\pi}', \vec{p}^* - \epsilon)$ whose revenue is at least $\alpha(\text{OPT} - \epsilon)$. We show that our approach runs in polynomial time in Section 5.

5 Runtime

Until now, we have only established that our algorithms are correct, up to maybe choosing the right parameters in WSO, which was deferred to this section. Here, we set these parameters appropriately and analyze the running times of all our algorithms. In particular, we show that all reduced forms required in Section 4 can be computed in polynomial time, and that both WSO from Section 3 and our revenue maximizing LP from Section 4 run in polynomial time.

Analyzing WSO from Section 3. We start with the appropriate choice of δ. The proof of the following lemma is in Appendix G.1.

LEMMA 5.1. *Let S be any subset of weight vectors in $[-1,1]^d$, b be the bit complexity of $\vec{\pi}$, and ℓ be an upper bound on the bit complexity of $\mathcal{A}(\vec{w})$ for all $\vec{w} \in [-1,1]^d$. Then if $\vec{\pi} \notin Conv(\{\mathcal{A}(\vec{w}) | \vec{w} \in S\})$, there exists a weight vector $\vec{w}^*$ such that $\vec{\pi} \cdot \vec{w}^* \geq \max_{\vec{w} \in S}\{\mathcal{A}(\vec{w}) \cdot \vec{w}^*\} + 4\delta$, where $\delta = 2^{-poly(d,\ell,b)}$ (does not depend on S).*

The requirement that δ is chosen appropriately only appears in the proof of Lemma 3.2. As Lemma 5.1 describes an appropriate choice of δ for the proof to be correct, we take $\delta = 2^{-\text{poly}(d,\ell,b)}$ in the definition of WSO.

Next we address the appropriate choice of N for the number of iterations used in WSO. This is stated in Corollary 5.1, and proved in Appendix G.1.

COROLLARY 5.1. *There exists some $N = poly(d, \ell, b)$ such that, if WSO has not found a feasible point after N iterations of the ellipsoid algorithm, the following polytope ($P(S)$) is empty:*

$$t - \vec{\pi} \cdot \vec{w} \leq -\delta;$$

$$t \geq \mathcal{A}(\vec{w}') \cdot \vec{w}, \ \forall \vec{w}' \in S;$$

$$\vec{w} \in [-1,1]^d;$$

where S is the set of weights $\vec{w}'$ such that WSO queried $\widehat{WSO}$ on $(t, \vec{w}')$ for some t during its execution, b is the bit complexity of $\vec{\pi}$, ℓ is an upper bound on the bit complexity of $\mathcal{A}(\vec{w})$ for all $\vec{w} \in [-1,1]^d$, and δ is chosen as in Lemma 5.1.

Note that Lemma 5.1 and Corollary 5.1 complete the description of WSO, and establish the truth of Lemma 3.2.

It remains to bound the running time of WSO. Let $rt_{\mathcal{A}}(x)$ be the running time of algorithm $\mathcal{A}$ on input whose bit complexity is x. With Lemma 5.1 and Corollary 5.1, we can bound the running time of WSO. This is stated below as Corollary 5.2 and proved in Appendix G.1.

COROLLARY 5.2. *Let b denote the bit complexity of $\vec{\pi}$ and ℓ be an upper bound of the bit complexity of $\mathcal{A}(\vec{w})$ for all $\vec{w} \in [-1,1]^d$. Then on input $\vec{\pi}$, WSO terminates in time $poly(d, \ell, b, rt_{\mathcal{A}}(poly(d, \ell, b)))$.*

Computing Reduced Forms. In Section 4 we need to use a possibly randomized social-welfare algorithm A (to which we have black-box access) to obtain an α-approximation algorithm $\mathcal{A}$ for optimizing any linear function $\vec{w} \cdot \vec{x}$ over $\vec{x} \in P = F(\mathcal{F}, \mathcal{D}')$, where $\mathcal{D}'$ is a (correlated across bidders) uniform distribution over $\text{poly}(n, T, 1/\epsilon)$ type profiles. We need to argue that for a given input $\vec{w} \in \mathbb{R}^T$ we can compute $\mathcal{A}(\vec{w}) \equiv R^A_{\mathcal{D}'}(\vec{w})$ in time polynomial in the description of $\vec{w}$ and the description of the distribution $\mathcal{D}'$. If A is randomized we cannot do this exactly, but we *do* get with high probability a good enough approximation for our purposes. We explain how to do this in Appendix G.2. The outcome is an algorithm $\mathcal{A}$, which has the following properties with probability at least $1 - \eta$, and for arbitrary choices of $\eta \in (0,1)$ and $\gamma \in (0, \alpha)$:

- for all $\vec{w}$ for which our algorithm from Section 4 may possibly query $\mathcal{A}$, $\mathcal{A}$ approximately optimizes the linear objective $\vec{w} \cdot \vec{x}$ over $\vec{x} \in F(\mathcal{F}, \mathcal{D}')$ to within a factor of $(\alpha - \gamma)$;
- the bit complexity of $\mathcal{A}(\vec{w})$ is always polynomial in the dimension T and the logarithm of the size, $\text{poly}(n, T, 1/\epsilon)$, of the support of $\mathcal{D}'$;

- on input $\vec{w}$ of bit complexity y, the running time of $\mathcal{A}$ is

$$rt_{\mathcal{A}}(y) = \text{poly}(n, T, \hat{\ell}, 1/\epsilon, \log 1/\eta, 1/\gamma, y) \cdot rt_A(\text{poly}(n, T, \hat{\ell}, \log 1/\epsilon, y)),$$

  where $rt_A(\cdot)$ represents the running time of A and $\hat{\ell}$ the bit complexity of the coordinates of the points in $\times_i T_i$.

Note that replacing α with $\alpha - \gamma$ in Section 4 does not affect our guarantees, except for a loss of a small amount in revenue and truthfulness, which can be made arbitrarily small with γ.

Analyzing the Revenue Optimizing LP. First we show that the WSO used in Section 4 as a proxy for a separation oracle for $F(\mathcal{F}, \mathcal{D}')$ runs in polynomial time. Recall that the dimension is $d = T$, the bit complexity of $\mathcal{A}(\vec{w})$ for any $\vec{w}$ can be bounded by $\ell = \text{poly}(n, T, \log 1/\epsilon)$, and that γ and η are constants used in the definition of $\mathcal{A}$. Hence, we immediately get the following corollary of Corollary 5.2.

COROLLARY 5.3. *Let b denote the bit complexity of $\vec{\pi}$. Then on input $\vec{\pi}$, WSO terminates in time*

$$poly(b, n, T, \hat{\ell}, 1/\epsilon, \log 1/\eta, 1/\gamma) \cdot rt_A(\text{poly}(n, T, \hat{\ell}, \log 1/\epsilon, b)),$$

where $\hat{\ell}$ is an upper bound on the bit complexity of the coordinates of the points in $\times_i T_i$.

Now that we have shown that WSO runs in polynomial time, we need to show that our revenue maximizing LP does as well. The proof of the following is in Appendix G.3.

LEMMA 5.2. *Let $\hat{\ell}$ denote an upper bound on the bit complexity of α, $v_{ij}(B)$ and $\Pr[t_i = B]$ for all i, j, B. Then the revenue maximizing LP (if we replace P with WSO)[7] terminates in time*

$$poly(n, T, \hat{\ell}, 1/\epsilon, \log 1/\eta, 1/\gamma) \cdot rt_A(\text{poly}(n, T, \hat{\ell}, \log 1/\epsilon)).$$

With this lemma we complete our proof that our algorithm from Section 4 is both correct and computationally efficient.

[7] See what we mean by "replacing P with WSO" in Section 4.

6 Formal Theorem Statements

In this section we provide our main theorem, formalizing Informal Theorem 1. In Appendix A, we also provide two extensions of our theorem to item-symmetric settings using the techniques of [17]. These extensions are Theorems A.1 and A.2 of Appendix A. In all cases, *the allocation rule of the mechanism output by our algorithm is a distribution over virtual implementations of the given social-welfare algorithm A.* Moreover, the mechanisms are ϵ-BIC and not truly-BIC, as we only know how to implement the target reduced forms exactly when consumers are sampled from $\mathcal{D}'$ (see discussion in Section 4). Theorems 6.1, A.1 and A.2 follow directly from Sections 3 through 5 in the same way that their corresponding theorems (Theorems 6 through 8) in Section 6 of [12] (arXiv version) follow, after replacing the separation oracle for $F(\mathcal{F}, \mathcal{D}')$ with WSO in the LP of Figure 2. In all theorem statements, $rt_A(x)$ denotes the runtime of algorithm A on inputs of bit complexity x.

THEOREM 6.1. *For all $\epsilon, \eta > 0$, all $\mathcal{D}$ of finite support in $[0,1]^{nm}$, and all $\mathcal{F}$, given black-box access to a (non-truthful) α-approximation algorithm, A, for finding the welfare-maximizing allocation in $\mathcal{F}$, there is a polynomial-time randomized approximation algorithm for MDMDP with the following properties: the algorithm obtains expected revenue $\alpha(OPT - \epsilon)$, with probability at least $1 - \eta$, in time polynomial in $\ell, n, T, 1/\epsilon, \log(1/\eta)$ and $rt_A(poly(\ell, n, T, \log 1/\epsilon, \log\log(1/\eta)))$, where ℓ is an upper bound on the bit complexity of the coordinates of the points in the support of $\mathcal{D}$, as well as of the probabilities assigned by $\mathcal{D}_1, \ldots, \mathcal{D}_m$ to the points in their support. The output mechanism is ϵ-BIC, and can be implemented in the same running time.*

We remark that we can easily modify Theorem 6.1 and its extensions (Theorems A.1 and A.2) to accommodate bidders with hard budget constraints. We simply add into the revenue-maximizing LP constraints of the form $p_i(\vec{v}_i) \leq B_i$, where B_i is bidder i's budget. It is easy to see that this approach works; this is addressed formally in [11, 12, 17].

References

[1] Saeed Alaei. Bayesian Combinatorial Auctions: Expanding Single Buyer Mechanisms to Many Buyers. In *the 52nd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2011.

[2] Saeed Alaei, Hu Fu, Nima Haghpanah, Jason Hartline, and Azarakhsh Malekian. Bayesian Optimal Auctions

via Multi- to Single-agent Reduction. In *the 13th ACM Conference on Electronic Commerce (EC)*, 2012.

[3] Moshe Babaioff, Ron Lavi, and Elan Pavlov. Single-value combinatorial auctions and implementation in undominated strategies. In *the 17th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2006.

[4] Ashwinkumar Badanidiyuru, Robert Kleinberg, and Yaron Singer. Learning on a budget: posted price mechanisms for online procurement. In *the 13th ACM Conference on Electronic Commerce (EC)*, 2012.

[5] Xiaohui Bei and Zhiyi Huang. Bayesian Incentive Compatibility via Fractional Assignments. In *the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2011.

[6] Sayan Bhattacharya, Gagan Goel, Sreenivas Gollapudi, and Kamesh Munagala. Budget Constrained Auctions with Heterogeneous Items. In *the 42nd ACM Symposium on Theory of Computing (STOC)*, 2010.

[7] Kim C. Border. Implementation of reduced form auctions: A geometric approach. *Econometrica*, 59(4):1175–1187, 1991.

[8] Kim C. Border. Reduced Form Auctions Revisited. *Economic Theory*, 31:167–181, 2007.

[9] Patrick Briest, Piotr Krysta, and Berthold Vöcking. Approximation techniques for utilitarian mechanism design. In *the 37th Annual ACM Symposium on Theory of Computing (STOC)*, 2005.

[10] Yang Cai and Constantinos Daskalakis. Extreme-Value Theorems for Optimal Multidimensional Pricing. In *the 52nd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2011.

[11] Yang Cai, Constantinos Daskalakis, and S. Matthew Weinberg. An Algorithmic Characterization of Multi-Dimensional Mechanisms. In *the 44th Annual ACM Symposium on Theory of Computing (STOC)*, 2012.

[12] Yang Cai, Constantinos Daskalakis, and S. Matthew Weinberg. Optimal Multi-Dimensional Mechanism Design: Reducing Revenue to Welfare Maximization. In *the 53rd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2012. http://arxiv.org/abs/1207.5518.

[13] Yang Cai and Zhiyi Huang. Simple and Nearly Optimal Multi-Item Auctions. In *the 24th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2013.

[14] Shuchi Chawla, Jason D. Hartline, and Robert D. Kleinberg. Algorithmic Pricing via Virtual Valuations. In *the 8th ACM Conference on Electronic Commerce (EC)*, 2007.

[15] Shuchi Chawla, Jason D. Hartline, David L. Malec, and Balasubramanian Sivan. Multi-Parameter Mechanism Design and Sequential Posted Pricing. In *the 42nd ACM Symposium on Theory of Computing (STOC)*, 2010.

[16] Yeon-Koo Che, Jinwoo Kim, and Konrad Mierendorff. Generalized Reduced-Form Auctions: A Network-Flow Approach. *University of Zürich, ECON-Working Papers*, 2011.

[17] Constantinos Daskalakis and S. Matthew Weinberg. Symmetries and Optimal Multi-Dimensional Mechanism Design. In *the 13th ACM Conference on Electronic Commerce (EC)*, 2012.

[18] Shaddin Dughmi and Tim Roughgarden. Black-box randomized reductions in algorithmic mechanism design. In *the 51st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2010.

[19] Martin Grötschel, László Lovász, and Alexander Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1(2):169–197, 1981.

[20] Martin Grötschel, Lászlo Lovász, and Alexander Schrijver. *Geometric Algorithms and Combinatorial Optimization*, volume 2 of *Algorithms and Combinatorics*. Springer, 1988.

[21] Sergiu Hart and Noam Nisan. Approximate Revenue Maximization with Multiple Items. In *the 13th ACM Conference on Electronic Commerce (EC)*, 2012.

[22] Jason D. Hartline, Robert Kleinberg, and Azarakhsh Malekian. Bayesian Incentive Compatibility via Matchings. In *the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2011.

[23] Jason D. Hartline and Brendan Lucier. Bayesian Algorithmic Mechanism Design. In *the 42nd ACM Symposium on Theory of Computing (STOC)*, 2010.

[24] Klaus Jansen. Approximate Strong Separation with Application in Fractional Graph Coloring and Preemptive Scheduling. In *the 19th Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, 2002.

[25] Robert Kleinberg and S. Matthew Weinberg. Matroid prophet inequalities. In *the 44th Annual ACM Symposium on Theory of Computing (STOC)*, 2012.

[26] A. M. Manelli and D. R. Vincent. Multidimensional Mechanism Design: Revenue Maximization and the Multiple-Good Monopoly. *Journal of Economic Theory*, 137(1):153–185, 2007.

[27] Roger B. Myerson. Optimal Auction Design. *Mathematics of Operations Research*, 6(1):58–73, 1981.

[28] Tim Roughgarden, Inbal Talgam-Cohen, and Qiqi Yan. Supply-limiting mechanisms. In *13th ACM Conference on Electronic Commerce (EC)*, 2012.

A Extensions of Theorem 6.1

This section contains extensions of Theorem 6.1 enabled by the techniques of [17].

THEOREM A.1. *For all $\epsilon, \eta > 0$, item-symmetric $\mathcal{D}$ of finite support in $[0,1]^{nm}$, item-symmetric $\mathcal{F}$,*[8] *and given black-box access to a (non-truthful) α-approximation algorithm, A, for finding the welfare-maximizing allocation in $\mathcal{F}$, there is a polynomial-*

[8] Distributions and feasibility constraints are item-symmetric if they are invariant under every item permutation.

time randomized approximation algorithm for MDMDP with the following properties: the algorithm obtains expected revenue $\alpha(OPT - \epsilon)$, with probability at least $1 - \eta$, in time polynomial in ℓ, $m, n^c, 1/\epsilon, \log(1/\eta)$ and $rt_A(poly(n^c, m, \log 1/\epsilon, \log\log(1/\eta), \ell))$, where $c = \max_{i,j} |\mathcal{D}_{ij}|$, and $|\mathcal{D}_{ij}|$ is the cardinality of the support of the marginal of $\mathcal{D}$ on bidder i and item j, and ℓ is as in the statement of Theorem 6.1. The output mechanism is ϵ-BIC, and can be implemented in the same running time.

THEOREM A.2. *For all $\epsilon, \eta, \delta > 0$, item-symmetric $\mathcal{D}$ supported on $[0,1]^{nm}$, item-symmetric $\mathcal{F}$, and given black-box access to a (non-truthful) α-approximation algorithm, A, for finding the welfare-maximizing allocation in $\mathcal{F}$, there is a polynomial-time randomized approximation algorithm for MDMDP with the following properties: If C is the maximum number of items that are allowed to be allocated simultaneously by $\mathcal{F}$, the algorithm obtains expected revenue $\alpha(OPT - (\sqrt{\epsilon} + \sqrt{\delta})C)$, with probability $1 - \eta$, in time polynomial in $m, n^{1/\delta}, 1/\epsilon, \log(1/\eta)$, and $rt_A(poly(n^{1/\delta}m, \log 1/\epsilon, \log\log 1/\eta))$. In particular, the runtime does* not *depend on $|\mathcal{D}|$ at all). The output mechanism is ϵ-BIC, and can be implemented in the same running time.*

REMARK 1. *The assumption that $\mathcal{D}$ is supported in $[0,1]^{mn}$ as opposed to some other bounded set is w.l.o.g., as we could just scale the values down by a multiplicative $v_{\max}$. This would cause the additive approximation error to be $\epsilon v_{\max}$. In addition, the point of the additive error in the revenue of Theorem A.2 is* not *to set ϵ, δ so small that they cancel out the factor of C, but rather to accept the factor of C as lost revenue. For "reasonable" distributions, the optimal revenue scales with C, so it is natural to expect that the additive loss should scale with C as well.*

B Appendix to Preliminaries

Here we provide two missing details from the Preliminaries: a formal definition of Bayesian Incentive Compatibility (BIC) and Individual Rationality (IR) for (possibly correlated) bidders.

DEFINITION 4. *[17](BIC/ϵ-BIC Mechanism) A direct mechanism M is called ϵ-BIC iff the following inequality holds for all bidders i and types $\tau_i, \tau_i' \in T_i$:*

$$\begin{aligned}\mathbb{E}_{t_{-i}\sim\mathcal{D}_{-i}(\tau_i)}&\left[U_i(\tau_i, M_i(\tau_i\ ;\ t_{-i}))\right] \\ &\geq \mathbb{E}_{t_{-i}\sim\mathcal{D}_{-i}(\tau_i)}\left[U_i(\tau_i, M_i(\tau_i'\ ;\ t_{-i}))\right] \\ &\quad - \epsilon v_{\max}\cdot\max\left\{1, \sum_j \pi_{ij}^M(\tau_i', \tau_i)\right\},\end{aligned}$$

where:

- *$U_i(B, M_i(C\ ;\ t_{-i}))$ denotes the utility of bidder i for the outcome of mechanism M if his true type is B, he reports C to the mechanism, and the other bidders report t_{-i};*
- *$v_{\max}$ is the maximum possible value of any bidder for any item in the support of the value distribution; and*
- *$\pi_{ij}^M(A, B)$ is the probability that item j is allocated to bidder i by mechanism M if bidder i reports type A to the mechanism, in expectation over the types of the other bidders, assuming they are drawn from $\mathcal{D}_{-i}(B)$ and report truthfully, as well as the mechanism's internal randomness.*

In other words, M is ϵ-BIC iff when a bidder i lies by reporting τ_i' instead of his true type τ_i, she does not expect to gain more than $\epsilon v_{\max}$ times the maximum of 1 and the expected number of items that she would receive by reporting τ_i' instead. A mechanism is called BIC iff it is 0-BIC.[9]

DEFINITION 5. *(IR/ϵ-IR) A direct mechanism M is called (interim) ϵ-IR iff the following inequality holds for all bidders i and types $\tau_i \in T_i$:*

$$\mathbb{E}_{t_{-i}\sim\mathcal{D}_{-i}(\tau_i)}\left[U_i(\tau_i, M_i(\tau_i\ ;\ t_{-i}))\right] \geq -\epsilon,$$

where $U_i(B, M_i(C\ ;\ t_{-i}))$ is as in Definition 4. A mechanism is said to be IR iff it is 0-IR.

C Input Model

We discuss two models for accessing a value distribution $\mathcal{D}$ over some known $\times_i T_i$, as well as what modifications are necessary, if any, to our algorithms to work with each model:

- **Exact Access:** We are given access to a sampling oracle as well as an oracle that exactly integrates the pdf of the distribution over a specified region.
- **Sample-Only Access:** We are given access to a sampling oracle and nothing else.

The presentation of the paper focuses on the first model. In this case, we can exactly evaluate the probabilities of events without any special care. If we have sample-only access to the distribution, some care is required. Contained in Appendix A of [17] is a sketch of the modifications necessary for all our results to apply with

[9] Strictly speaking, the definition of BIC in [17] is the same but without taking a max with 1. We are still correct in applying their results with this definition because any mechanism that is considered ϵ-BIC by [17] is certainly considered ϵ-BIC by this definition. We basically call a mechanism ϵ-BIC if either the definition in [5, 22, 23] ($\epsilon v_{\max}$) or [17] ($\epsilon v_{\max}\sum_j \pi_{ij}(\tau_i')$) holds.

sample-only access. The sketch is given for the item-symmetric case, but the same approach will work in the asymmetric case. Simply put, repeated sampling will yield some distribution $\mathcal{D}'$ that is very close to $\mathcal{D}$ with high probability. If the distributions are close enough, then a solution to the MDMDP for $\mathcal{D}'$ is an approximate solution for $\mathcal{D}$. The error in approximating $\mathcal{D}$ is absorbed into the additive error in both revenue and truthfulness.

D Additive Dimension

Here we discuss the notion of additive dimension and show some interesting examples of settings with low additive dimension. Consider two settings, both with the same *possible type-space* for each bidder, $\hat{T}_i$ (i.e. $\hat{T}_i$ is the entire set of types that the settings model, $T_i \subseteq \hat{T}_i$ is the set of types that will ever be realized for the given distribution. As a concrete example, $\hat{T}_i = \mathbb{R}^n$ for additive settings.): the first is the "real" setting, with the actual items and actual bidder valuations. The real setting has n items, m bidders, feasibility constraints $\mathcal{F}$, and valuation functions $V_{i,B}(S) : \mathcal{F} \to \mathbb{R}$ for all $i, B \in \hat{T}_i$ that map $S \in \mathcal{F}$ to a value of bidder i of type B for the allocation of items S. The second is the "meta" setting, with meta-items. The meta-setting has d meta-items, m bidders, feasibility constraints $\mathcal{F}'$, and valuation functions $V'_{i,B}(S') : \mathcal{F}' \to \mathbb{R}$ for all $i, B \in \hat{T}_i$ that map $S' \in \mathcal{F}'$ to the value of bidder i of type B for the allocation of meta-items S'. We now define what it means for a meta-setting to faithfully represent the real setting.

Definition 6. *A meta-setting is equivalent to a real setting if there is a mapping from $\mathcal{F}$ to $\mathcal{F}'$, g, and another from $\mathcal{F}'$ to $\mathcal{F}$, h, such that $V_{i,B}(S) = V'_{i,B}(g(S))$, and $V'_{i,B}(S') = V_{i,B}(h(S'))$ for all $i, B \in \hat{T}_i, S \in \mathcal{F}, S' \in \mathcal{F}'$.*

When two settings are equivalent, there is a natural mapping between mechanisms in each setting. Specifically, let M be any mechanism in the real setting. Then in the meta-setting, have M' run M, and if M selects allocation S of items, M' selects the allocation $g(S)$ of meta-items and charges exactly the same prices. It is clear that when bidders are sampled from the same distribution, M is BIC/IC/IR if and only if M' is as well. It is also clear that M and M' achieve the same expected revenue. The mapping in the other direction is also obvious, just use h. We now define the additive dimension of an auction setting.

Definition 7. *The* additive dimension *of an auction setting is the minimum d such that there is an equivalent (by Definition 6) meta-setting with additive bidders and d meta-items (i.e. due to the feasibility constraints, all bidders valuations can be models as additive over their values for each meta-item).*

In Section 1, we observed that all of our results also apply to settings with additive dimension d after multiplying the runtimes by a poly(d) factor. This is because a black-box algorithm for approximately maximizing welfare in the real setting is also a black-box algorithm for approximately maximizing welfare in the meta-setting (just apply g to whatever the algorithm outputs). So if we have black-box access to a social welfare algorithm for the real setting, we have black-box access to a social welfare algorithm for the meta-setting. As the meta-setting is additve, all of our techniques apply. We then just apply h at the end and obtain a feasible allocation in the real setting.

We stress that important properties of the setting are not necessarily preserved under the transformation from the real to meta setting. Importantly, when the real setting is downwards closed, this is *not* necessarily true for the meta-setting. The user of this transformation should be careful of issues arising due to negative weights if the desired meta-setting is not downwards-closed.

Respecting the required care, we argued in Section 1 that single-minded combinatorial auctions had additive dimension 1 (and the meta-setting is still downwards-closed, and therefore can accommodate negative values). Now we will show that two other natural models have low additive dimension, and that their corresponding meta-settings are downwards-closed. The discussions below are not intended to be formal proofs. The point of this discussion is to show that interesting non-additive settings have low additive dimension (via meta-settings where approximation algorithms can accommodate negative values) and can be solved using our techniques.

D.1 d-minded Combinatorial Auctions A d-minded combinatorial auction setting is where each bidder i has at most d (public) subsets of items that they are interested in, and a (private) value v_{ij} for receiving the j^{th} subset in their list, S_{ij}, and value 0 for receiving any other subset. Such bidders are clearly not additive over their value for the items, but have additive dimension d. Specifically, make d meta-items. Define $g(S)$ so that if bidder i receives subset S_{ij} in S, they receive item j in $g(S)$. Define $h(S')$ so that if bidder i receives item j in S', they receive the subset of items S_{ij} in $h(S')$. Also define $\mathcal{F}'$ so that an allocation is feasible iff it assigns each bidder at most 1 meta-item, and when bidder i is assigned meta-item j_i, the sets $\{S_{ij_i} | i \in [m]\}$ are pairwise disjoint. Finally, set

$V'_{i,B}(j) = V_{i,B}(S_{ij})$. Then it is clear that these two settings are equivalent. It is also clear that bidders are additive in the meta-setting as they are unit-demand (i.e. they can never feasibly receive more than one item). Therefore, d-minded Combinatorial Auctions have additive dimension d, and any (not necessarily truthful) α-approximation algorithm for maximizing welfare implies a (truthful) $(\alpha - \epsilon)$-approximation algorithm for maximizing revenue whose runtime is $\text{poly}(d, T, 1/\epsilon, b)$. It is also clear that the meta-setting is downwards-closed, and therefore all (not necessarily truthful) α-approximation algorithms for maximizing welfare can accommodate negative values.

D.2 Combinatorial Auctions with Symmetric Bidders. A bidder is symmetric if their value $V_{i,B}(S) = V_{i,B}(U)$ whenever $|S| = |U|$ (i.e. bidders only care about the cardinality of sets they receive). Such bidders (with the extra constraint of submodularity) are studied in [4]. Such bidders are again clearly not additive over their values for the items, but have additive dimension n. Specifically, make n meta-items. Define $g(S)$ to assign bidder i item j if they received exactly j items in S. Define $h(S')$ to assign bidder i exactly j items if they were awarded item j in S' (it doesn't matter in what order the items are handed out, lexicographically works). Also define $\mathcal{F}'$ so that an allocation is feasible iff it assigns each bidder at most 1 meta-item and when bidder i is assigned meta-item j_i, we have $\sum_i j_i \leq n$. Finally, set $V'_{i,B}(j) = V_{i,B}(S)$ where S is any set with cardinality j. It is again clear that the two settings are equivalent. It is also clear that the meta-setting is unit-demand, so bidders are again additive. Therefore, combinatorial auctions with symmetric bidders have additive dimension n, and any (not necessarily truthful) α-approximation algorithm for maximizing welfare implies a (truthful) $(\alpha - \epsilon)$-approximation algorithm for maximizing revenue whose runtime is $\text{poly}(n, T, 1/\epsilon, b)$. It is also clear that the meta-setting is downwards-closed, and therefore all (not necessarily truthful) α-approximation algorithms for maximizing welfare can accomodate negative values.

In addition, we note here that it is possible to exactly optimize welfare in time $\text{poly}(n, m)$ for symmetric bidders (even with negative, not necessarily submodular values) using a simple dynamic program. We do not describe the algorithm as that is not the focus of this work. We make this note to support that this is another interesting non-additive setting that can be solved using our techniques.

E Omitted Proofs from Section 3

E.1 Omitted Proofs from Section 3.1. *Proof of Lemma 3.1:* If $\vec{\pi} \in P_0$, then $\vec{\pi} \cdot \vec{w} \leq \alpha \max_{x \in P}\{x \cdot \vec{w}\} \leq \mathcal{A}(\vec{w}) \cdot \vec{w}$ for all $\vec{w} \in [-1, 1]^d$, since $\mathcal{A}$ is an α-approximation algorithm. Therefore, $\vec{\pi} \in P_1$ as well. So $P_0 \subseteq P_1$.

Recall now that a point $\vec{\pi}$ is in the convex hull of S if and only if for all $\vec{w} \in [-1, 1]^d$, there exists a point $\vec{x}(\vec{w}) \in S$ such that $\vec{\pi} \cdot \vec{w} \leq \vec{x}(\vec{w}) \cdot \vec{w}$. If $\vec{\pi} \in P_1$, then we may simply let $\vec{x}(\vec{w}) = \mathcal{A}(\vec{w})$ to bear witness that $\vec{\pi} \in P_2(A, \mathcal{D})$. □

E.2 Omitted Proofs from Section 3.2. *Proof of Fact 3.1:* Any hyperplane output by WSO is of the form $\vec{w}^* \cdot \vec{\pi} \leq t^*$. Because $\vec{w}^*, t^*$ was accepted by $\widehat{WSO}$, we must have $t^* \geq \mathcal{A}(\vec{w}^*) \cdot \vec{w}^*$. As every point in P_1 satisfies $\vec{\pi} \cdot \vec{w}^* \leq \mathcal{A}(\vec{w}^*) \cdot \vec{w}^* \leq t^*$, we get that $P_1 \subseteq Q$. □

Proof of Fact 3.2: In order for WSO to reject $\vec{\pi}$, its internal ellipsoid algorithm that uses $\widehat{WSO}$ must find some feasible point $(t^*, \vec{w}^*)$. As $\widehat{WSO}$ accepts $(t^*, \vec{w}^*)$, such a point clearly satisfies the following two equations:

$$t^* < \vec{\pi} \cdot \vec{w}^*$$

$$t^* \geq \mathcal{A}(\vec{w}^*) \cdot \vec{w}^*$$

Together, this implies that $\vec{\pi} \cdot \vec{w}^* > \mathcal{A}(\vec{w}^*) \cdot \vec{w}^*$, so $\vec{\pi} \notin P_1$. □

Proof of Corollary 3.1: By Fact 3.2, whenever WSO rejects $\vec{\pi}$, $\vec{\pi} \notin P_1$. By Fact 3.1, any halfspace output by WSO contains P_1. This is all that is necessary in order for WSO to act as a valid separation oracle for P_1 when it rejects $\vec{\pi}$. □

Proof of Lemma 3.2: Define the polytope $P(S)$ as the set of $t, \vec{w}$ that satisfy the following inequalities:

$$t - \vec{\pi} \cdot \vec{w} \leq -\delta$$

$$t \geq \mathcal{A}(\vec{w}') \cdot \vec{w}, \ \forall \vec{w}' \in S$$

$$\vec{w} \in [-1, 1]^d$$

We first claim that if $\vec{\pi} \notin \mathit{Conv}(\{\mathcal{A}(\vec{w}) | \vec{w} \in S\})$, then $P(S)$ is non-empty. This is because when $\vec{\pi} \notin \mathit{Conv}(\{\mathcal{A}(\vec{w}) | \vec{w} \in S\})$, there is some weight vector $\vec{w}^* \in [-1, 1]^d$ such that $\vec{w}^* \cdot \vec{\pi} > \max_{\vec{w}' \in S}\{\vec{w}^* \cdot \mathcal{A}(\vec{w}')\}$. For appropriately chosen δ (Lemma 5.1 in Section 5 provides one), there is also a $\vec{w}^*$ such that $\vec{w}^* \cdot \vec{\pi} \geq \max_{\vec{w}' \in S}\{\vec{w}^* \cdot \mathcal{A}(\vec{w}')\} + \delta$. Set $t^* := \max_{\vec{w}' \in S}\{\vec{w}^* \cdot \mathcal{A}(\vec{w}')\}$ and consider the point $(t^*, \vec{w}^*)$. As t^* is larger than $\mathcal{A}(\vec{w}') \cdot \vec{w}^*$ for all $\vec{w}' \in S$ by definition, and $\vec{w}^* \in [-1, 1]^d$ by definition, we have found a point in $P(S)$.

Now, consider that in the execution of $WSO(\vec{\pi})$, $\widehat{WSO}$ outputs several halfspaces. As S is exactly the set of weights $\vec{w}$ that WSO queries to $\widehat{WSO}$, these are exactly the halfspaces:

$$t \geq \mathcal{A}(\vec{w}') \cdot \vec{w}, \ \forall \vec{w}' \in S$$

During the execution of $WSO(\vec{\pi})$, other halfspaces may be learned not from $\widehat{WSO}$, but of the form $t - \vec{\pi} \cdot \vec{w} \leq -\delta$ or $-1 \leq w_i$, $w_i \leq 1$ for some $i \in [d]$. Call the polytope defined by the intersection of all these halfspaces $P(WSO)$. As all of these halfspaces contain $P(S)$, it is clear that $P(WSO)$ contains $P(S)$.

Now we just need to argue that if N is sufficiently large, and $WSO(\vec{\pi})$ could not find a feasible point in N iterations, then $P(S)$ is empty. Corollary 5.1 in Section 5 provides an appropriate choice of N. Basically, if $P(S)$ is non-empty, we can lower bound its volume with some value V (independent of S). If $N = \text{poly}(\log(1/V))$, then the volume of the ellipsoid containing $P(WSO)$ after N iterations will be strictly smaller than V. As $P(WSO)$ contains $P(S)$, this implies that $P(S)$ is empty. Therefore, we may conclude that $\vec{\pi} \in Conv(\{\mathcal{A}(\vec{w}) | \vec{w} \in S\})$. □

F Omitted Figures and Proofs from Section 4

Proof of Lemma 4.1: Let $(\vec{\pi}^*, \vec{p}^*)$ denote the reduced form output by the LP in Figure 2. Then we claim that the reduced form $(\alpha\vec{\pi}^*, \alpha\vec{p}^*)$ is a feasible solution after replacing $F(\mathcal{F}, \mathcal{D}')$ with P_0. It is clear that this mechanism is still IR and BIC, as we have simply multiplied both sides of every incentive constraint by α. It is also clear that $\alpha\vec{\pi}^* \in P_0$ by definition. As we have multiplied all of the payments by α, and the original LP had revenue $\text{OPT} - \epsilon$ [12], it is clear that revenue of the reduced form output in the new LP is at least $\alpha(\text{OPT} - \epsilon)$. □

Proof of Proposition 4.1: Let Q_0 be the set of $(\vec{\pi}, \vec{p})$ that satisfy $\vec{\pi} \in P_0$, the BIC and IR constraints, as well as the revenue constraint $\sum_{i=1}^{m} \sum_{\vec{v}_i \in T_i} \Pr[t_i = \vec{v}_i] \cdot p_i(\vec{v}_i) \geq x$. As $x \leq Rev(P_0)$, we know that there is some feasible point $(\vec{\pi}^*, \vec{p}^*) \in Q_0$. Therefore, the ellipsoid algorithm using a valid separation oracle for Q_0 and the correct parameters will find a feasible point.

Now, what is the difference between a valid separation oracle for Q_0 and WSO' as used in Proposition 4.1? A separation oracle for Q_0 first checks the BIC, IR, and revenue constraints, then executes a true separation oracle for P_0. WSO' first checks the BIC, IR, and revenue constraints, then executes WSO. So let us assume for contradiction that the Ellipsoid using WSO' outputs infeasible, but Q_0 is non-empty. It has to be then that WSO rejected every point that was queried to it.[10] However, Corollary 3.1 guarantees that when rejecting points, WSO acts as a valid separation oracle for P_0 (i.e. provides a valid hyperplane separating $\vec{\pi}$ from P_0). As the only difference between a separation oracle for Q_0 and WSO' was the use of WSO, and WSO acted as a valid separation oracle for P_0, this means that in fact WSO' behaved as a valid separation oracle for Q_0. So we ran the ellipsoid algorithm using a valid separation oracle for Q_0 with the correct parameters, but output "infeasible" when Q_0 was non-empty, contradicting the correctness of the ellipsoid algorithm.

Therefore, whenever Q_0 is non-empty, WSO' must find a feasible point. As Q_0 is non-empty whenever $x \leq Rev(P_0)$, this means that WSO' will find a feasible point whenever $x \leq Rev(P_0)$, proving the proposition.□

Proof of Corollary 4.1: Consider running the LP of Figure 2 with WSO. The optimization version of the ellip-

Near-Optimal LP:

Variables:

- $p_i(\vec{v}_i)$, for all bidders i and types $\vec{v}_i \in T_i$, denoting the expected price paid by bidder i when reporting type $\vec{v}_i$ over the randomness of the mechanism and the other bidders' types.
- $\pi_{ij}(\vec{v}_i)$, for all bidders i, items j, and types $\vec{v}_i \in T_i$, denoting the probability that bidder i receives item j when reporting type $\vec{v}_i$ over the randomness of the mechanism and the other bidders' types.

Constraints:

- $\vec{\pi}_i(\vec{v}_i) \cdot \vec{v}_i - p_i(\vec{v}_i) \geq \vec{\pi}_i(\vec{w}_i) \cdot \vec{v}_i - p_i(\vec{w}_i)$, for all bidders i, and types $\vec{v}_i, \vec{w}_i \in T_i$, guaranteeing that the reduced form mechanism $(\vec{\pi}, \vec{p})$ is BIC.
- $\vec{\pi}_i(\vec{v}_i) \cdot \vec{v}_i - p_i(\vec{v}_i) \geq 0$, for all bidders i, and types $\vec{v}_i \in T_i$, guaranteeing that the reduced form mechanism $(\vec{\pi}, \vec{p})$ is individually rational.
- $\vec{\pi} \in F(\mathcal{F}, \mathcal{D}')$, guaranteeing that the reduced form $\vec{\pi}$ is feasible for $\mathcal{D}'$.

Maximizing:

- $\sum_{i=1}^{m} \sum_{\vec{v}_i \in T_i} \Pr[t_i = \vec{v}_i] \cdot p_i(\vec{v}_i)$, the expected revenue **when played by bidders sampled from the true distribution** $\mathcal{D}$.

Figure 2: An LP with near-optimal revenue.

[10] In particular, even if Ellipsoid deems the feasible region lower-dimensional, and continues in a lower-dimensional space, etc., then still if the final output of Ellipsoid is infeasible, then all points that it queried to WSO were rejected.

soid algorithm will do a binary search on possible values for the objective function and solve a separate feasibility subproblem for each. Proposition 4.1 guarantees that on every feasibility subproblem with $x \leq Rev(P_0)$, the ellipsoid algorithm will find a feasible point. Therefore, the binary search will stop at some value $x^* \geq Rev(P_0)$, and we get that $Rev(WSO) \geq Rev(P_0)$. □

Proof of Corollary 4.3: Because $\vec{\pi}^*$ is output, we have $WSO(\vec{\pi}^*) =$ "yes." Lemma 3.2 tells us that $\vec{\pi}^*$ is therefore in the convex hull of $\{R^A_{\mathcal{D}'}(\vec{w})|\vec{w} \in S\}$. As a convex combination of reduced forms can be implemented as a distribution over the allocation rules that implement them, we have proven the corollary. □

F.1 The Solution is ϵ-BIC. Here we provide the omitted informal argument from Section 4. Again, the reader is referred to [12] for a formal proof. Recall that we are trying to show that the distribution over virtual implementations of A used to implement $\vec{\pi}^*$ when bidders are sampled from $\mathcal{D}'$ implements some reduced form $\vec{\pi}'$ when bidders are sampled from $\mathcal{D}$ satisfying $|\vec{\pi}^* - \vec{\pi}'|_1 \leq \epsilon$. This suffices to guarantee that our mechanism computed in Section 4 is ϵ-BIC.

Let x denote the number of samples taken for $\mathcal{D}'$. There are two steps in the argument: the first is showing that for a fixed allocation rule, the probability that its reduced form when bidders are sampled from $\mathcal{D}'$ is within ϵ in ℓ_1 distance of the reduced form when bidders are sampled from $\mathcal{D}$ approaches 1 exponentially fast in x. This can be done by a simple Chernoff bound. The second is showing that, before actually sampling $\mathcal{D}'$, but after choosing how many samples to take, the number of weights $\vec{w}$ that could possibly ever be queried to $\widehat{WSO}$ grows like $2^{\text{poly}(\log x)}$. This can be done by reasoning about bit complexities maintained by the ellipsoid algorithm, and the fact that the bit complexity of $\mathcal{D}'$ grows logarithmically in x. With both facts, we can then take a union bound over the entire set of weights that will ever be possibly queried to $\widehat{WSO}$ and get that with high probability, the reduced forms of all the allocation rules corresponding to these weights are ϵ-close under $\mathcal{D}$ and $\mathcal{D}'$. If this holds for all these allocation rules simultaneously, then it clearly also holds for any distribution over them (in particular, for $\vec{\pi}^*$). This entire argument appears formally in [12] for the case where the LP of Figure 2 uses $F(\mathcal{F}, \mathcal{D}')$ rather than WSO, and is nearly identical.

G Omitted Proofs from Section 5

G.1 Omitted Proofs on the Runtime of WSO from Section 5. *Proof of Lemma 5.1:* Consider the polytope $P'(S)$ with respect to variables t' and $\vec{w}'$ that is the intersection of the following half-spaces (similar to $P(S)$ from the proof of Lemma 3.2):

$$t' \leq d$$
$$t' \geq \mathcal{A}(\vec{w}) \cdot \vec{w}', \ \forall \vec{w} \in S$$
$$\vec{w}' \in [-1, 1]^d$$

If $\vec{\pi} \notin Conv(\{\mathcal{A}(\vec{w})|\vec{w} \in S\})$, there exists some weight vector $\vec{w}'$ such that $\vec{\pi} \cdot \vec{w}' > \max_{\vec{w} \in S}\{\mathcal{A}(\vec{w}) \cdot \vec{w}'\}$. This bears witness that there is a point in $P'(S)$ satisfying $\vec{\pi} \cdot \vec{w}' > t'$. If such a point exists, then clearly we may take $(\vec{w}', t')$ to be a corner of $P'(S)$ satisfying the same inequality. As the bit complexity of every halfspace defining $P'(S)$ is $\text{poly}(d, \ell)$, the corner also has bit complexity $\text{poly}(d, \ell)$. Therefore, if $t' - \vec{\pi} \cdot \vec{w}' < 0$, $t' - \vec{\pi} \cdot \vec{w}' \leq -4\delta$, for some $\delta = 2^{-\text{poly}(d,\ell,b)}$. □

LEMMA G.1. *Let S be any subset of weights. Let also b be the bit complexity of $\vec{\pi}$, and ℓ an upper bound on the bit complexity of $\mathcal{A}(\vec{w})$ for all $\vec{w}$. Then, if we choose δ as prescribed by Lemma 5.1, the following polytope ($P(S)$) is either empty, or has volume at least $2^{-poly(d,\ell,b)}$:*

$$t - \vec{\pi} \cdot \vec{w} \leq -\delta$$
$$t \geq \mathcal{A}(\vec{w}') \cdot \vec{w}, \ \forall \vec{w}' \in S$$
$$\vec{w} \in [-1, 1]^d$$

Proof of Lemma G.1: First, it will be convenient to add the vacuous constraint $t \leq d$ to the definition of the polytope. It is vacuous because it is implied by the existing constraints,[11] but useful for the analysis. Define $P'(S)$ by removing the first constraint. That is, $P'(S)$ is the intersection of the following halfspaces (this is the same as $P'(S)$ from the proof of Lemma 5.1):

$$t \leq d$$
$$t \geq \mathcal{A}(\vec{w}') \cdot \vec{w}, \ \forall \vec{w}' \in S$$
$$\vec{w} \in [-1, 1]^d$$

If there is a point in $P(S)$, then there is some point in $P'(S)$ satisfying $t - \vec{\pi} \cdot \vec{w} \leq -\delta$. If such a point exists, then clearly there is also a corner of $P'(S)$ satisfying $t - \vec{\pi} \cdot \vec{w} \leq -\delta$. Call this corner $(t^*, \vec{w}^*)$. Recall that δ was chosen in the proof of Lemma 5.1 so that we are actually guaranteed $t - \vec{\pi} \cdot \vec{w} \leq -4\delta$. Therefore, the point $(t^*/2, \vec{w}^*/2)$ is also clearly in $P(S)$, and satisfies $t - \vec{\pi} \cdot \vec{w} \leq -2\delta$.

Now, consider the box $B = [\frac{t^*}{2} + \frac{\delta}{2}, \frac{t^*}{2} + \frac{3\delta}{4}] \times (\times_{i=1}^d [\frac{w_i^*}{2}, \frac{w_i^*}{2} + \frac{\delta}{4d}])$. We claim that $B \subseteq P(S)$. Let

[11] Since $P \in [-1, 1]^d$, WLOG we can also assume $\vec{\pi} \in [-1, 1]^d$, thus from the constraints of $P(S)$ it follows that $t \leq d$.

$(t, \vec{w})$ denote an arbitrary point in B. It is clear that we have $\vec{w} \in [-1,1]^d$, as we had $\vec{w}^*/2 \in [-1/2, 1/2]^d$ to start with. As each coordinate of $\vec{\pi}$ and $\mathcal{A}(\vec{w}')$ for all $\vec{w}'$ is in $[-1,1]$, it is easy to see that:

$$(\vec{w}^*/2) \cdot \vec{\pi} - \frac{\delta}{4} \leq \vec{w} \cdot \vec{\pi} \leq (\vec{w}^*/2) \cdot \vec{\pi} + \frac{\delta}{4},$$

and for all $\vec{w}' \in S$,

$$(\vec{w}^*/2) \cdot \mathcal{A}(\vec{w}') - \frac{\delta}{4} \leq \vec{w} \cdot \mathcal{A}(\vec{w}') \leq (\vec{w}^*/2) \cdot \mathcal{A}(\vec{w}') + \frac{\delta}{4}.$$

As we must have $t \geq \frac{t^*}{2} + \frac{\delta}{2}$, and we started with $t^* \geq \vec{w}^* \cdot \mathcal{A}(\vec{w}')$ for all $\vec{w}' \in S$, it is clear that we still have $t \geq \vec{w} \cdot \mathcal{A}(\vec{w}')$ for all $\vec{w}' \in S$. Finally, since we started with $t^*/2 - \vec{\pi} \cdot \vec{w}^*/2 \leq -2\delta$, and $t \leq t^*/2 + \frac{3\delta}{4}$, we still have $t - \vec{\pi} \cdot \vec{w} \leq -\delta$.

Now, we simply observe that the volume of B is $\frac{\delta^{d+1}}{d^d 4^{d+1}}$, which is $2^{-\text{poly}(d,\ell,b)}$. Therefore, if $P(S)$ is non-empty, it contains this box B, and therefore has volume at least $2^{-\text{poly}(d,\ell,b)}$. □

Proof of Corollary 5.1: By Lemma G.1, if $P(S)$ is non-empty, $P(S)$ has volume at least some $V = 2^{-\text{poly}(d,\ell,b)}$. Since $P \subseteq [-1,1]^d$, the starting ellipsoid in the execution of WSO can be taken to have volume $2^{O(d)}$. As the volume of the maintained ellipsoid shrinks by a multiplicative factor of at least $1 - \frac{1}{\text{poly}(d)}$ in every iteration of the ellipsoid algorithm, after some $N = \text{poly}(d, \ell, b)$ iterations, we will have an ellipsoid with volume smaller than V that contains $P(S)$ (by the proof of Lemma 3.2), a contradiction. Hence $P(S)$ must be empty, if we use the N chosen above for the definition of WSO and the ellipsoid algorithm in the execution of WSO does not find a feasible point after N iterations. □

Proof of Corollary 5.2: By the choice of N in Corollary 5.1, WSO only does $\text{poly}(d, \ell, b)$ iterations of the ellipsoid algorithm. Note that the starting ellipsoid can be taken to be the sphere of radius $\sqrt{d}$ centered at $\vec{0}$, as $P \subseteq [-1,1]^d$. Moreover, the hyperplanes output by $\widehat{WSO}$ have bit complexity $O(\ell)$, while all other hyperplanes that may be used by the ellipsoid algorithm have bit complexity $\text{poly}(d, \ell, b)$ given our choice of δ. So by [20], $\widehat{WSO}$ will only be queried at points of bit complexity $\text{poly}(d, \ell, b)$, and every such query will take time $\text{poly}(\text{poly}(d, \ell, b), rt_{\mathcal{A}}(\text{poly}(d, \ell, b)))$ as it involves checking one inequality for numbers of bit complexity $\text{poly}(d, \ell, b)$ and making one call to $\mathcal{A}$ on numbers of bit complexity $\text{poly}(d, \ell, b)$. Therefore, WSO terminates in the promised running time. □

G.2 Computing Reduced Forms of (Randomized) Allocation Rules. For distributions that are explicitly represented (such as $\mathcal{D}'$), it is easy to compute the reduced form of a deterministic allocation rule: simply iterate over every profile in the support of $\mathcal{D}'$, run the allocation rule, and see who receives what items. For randomized allocation rules, this is trickier as computing the reduced form exactly would require enumerating over the randomness of the allocation rule. One approach is to approximate the reduced form. This approach works, but is messy to verify formally, due to the fact that the bit complexity of reduced forms of randomized allocations takes effort to bound. The technically cleanest approach is to get our hands on a deterministic allocation rule instead.

Let A be a randomized allocation rule that obtains an α-fraction of the maximum welfare in expectation. Because the welfare of the allocation output by A cannot be larger than the maximum welfare, the probability that A obtains less than an $(\alpha - \gamma)$-fraction of the maximum welfare is at most $1 - \gamma$. So let A' be the allocation rule that runs several independent trials of A and chooses (of the allocations output in each trial) the one with maximum welfare. If the number of trials is x/γ, we can guarantee that A' obtains at least an $(\alpha - \gamma)$-fraction of the maximum welfare with probability $1 - e^{-x}$. From this it follows that, if $O((\ell+\tau)/\gamma)$ independent trials of A are used for A', then A' obtains an $(\alpha - \gamma)$-fraction of the maximum welfare for *all* input vectors of bit complexity ℓ, with probability at least $1 - 2^{-\tau}$. This follows by taking a union bound over all 2^ℓ possible vectors of bit complexity ℓ. For ℓ, τ to be determined later, we fix the randomness used by A' in running A ahead of time so that A' is a deterministic algorithm. Define $\mathcal{A}'$ using A' in the same way that $\mathcal{A}$ is defined using A.

As A' is a deterministic algorithm and $\mathcal{D}'$ a uniform distribution over $\text{poly}(n, T, 1/\epsilon)$ profiles, we can compute $R_{\mathcal{D}'}^{\mathcal{A}'}(\vec{w})$ for a given $\vec{w}$ by enumerating over the support of $\mathcal{D}'$ as described above. The resulting $R_{\mathcal{D}'}^{\mathcal{A}'}(\vec{w})$ has bit complexity polynomial in the dimension T and the logarithm of $\text{poly}(n, T, 1/\epsilon)$.[12]

Now let us address the choice of ℓ. We basically want to guarantee the following. Suppose that we use $\mathcal{A}'$ inside WSO. We want to guarantee that $\mathcal{A}'$ will work well for any vector $\vec{w}$ that our algorithm will possibly ask $\mathcal{A}'$.[13] We argue in the proof of

[12] This is because the probability that bidder i gets item j conditioned on being type B is just the number of profiles in the support of $\mathcal{D}'$ where $t_i = B$ and bidder i receives item j divided by the number of profiles where $t_i = B$. The definition of $\mathcal{D}'$ (see [12]) makes sure that the latter is non-zero.

[13] Namely, we want that $\mathcal{A}'$ approximately optimizes the linear

Lemma 5.2 that, regardless of the bit complexity of the hyperplanes output by WSO, throughout the execution of our algorithm WSO will only be queried on points of bit complexity $\text{poly}(n, T, \hat{\ell}, \log 1/\epsilon)$, where $\hat{\ell}$ is the bit complexity of the coordinates of the points in $\times_i T_i$. From Corollary 5.2 it follows then that $\mathcal{A}'$ will only be queried on inputs of bit complexity $\text{poly}(n, T, \hat{\ell}, \log 1/\epsilon)$. This in turns implies that A' will only be queried on inputs of bit complexity $\text{poly}(n, T, \hat{\ell}, \log 1/\epsilon)$. So setting ℓ to some $\text{poly}(n, T, \hat{\ell}, \log 1/\epsilon)$ guarantees that A' achieves an $(\alpha-\gamma)$-fraction of the maximum welfare, for all possible inputs it may be queried simultaneously, with probability at least $1 - 2^{-\tau}$.

Finally, for every input $\vec{w}$ of bit complexity x, the running time of $\mathcal{A}'$ is polynomial in x, the support size and the bit complexity of $\mathcal{D}'$ (which is $\text{poly}(n, T, \hat{\ell}, 1/\epsilon)$), and the running time of A' on inputs of bit complexity $\text{poly}(n, T, \hat{\ell}, \log 1/\epsilon, x)$. The latter is just a factor of $\text{poly}(n, T, \hat{\ell}, \log 1/\epsilon, \tau, 1/\gamma)$ larger than that of A on inputs of bit complexity $\text{poly}(n, T, \hat{\ell}, \log 1/\epsilon, x)$. Overall,

$$rt_{\mathcal{A}'}(x) = \text{poly}(n, T, \hat{\ell}, 1/\epsilon, \tau, 1/\gamma, x) \cdot rt_A(\text{poly}(n, T, \hat{\ell}, \log 1/\epsilon, x)).$$

G.3 Omitted Proofs on the Runtime of the Revenue-Maximizing LP from Section 5. *Proof of Lemma 5.2:* Ignoring computational efficiency, we can use the construction of [12] to build a separation oracle for P_0, defined as in Section 4. Suppose that we built this separation oracle, SO, and used it to solve the LP of Figure 2 with P_0 in place of $F(\mathcal{F}, \mathcal{D}')$ using the ellipsoid algorithm. It follows from [20] that the ellipsoid algorithm using SO would terminate in time polynomial in $n, T, \hat{\ell}, \log 1/\epsilon$ and the running time of SO on points of bit complexity $\text{poly}(n, T, \hat{\ell}, \log 1/\epsilon)$.[14] As we are running exactly this algorithm (i.e. with the same parameters and criterion for deeming the feasible region lower-dimensional), except replacing the separation oracle for P_0 with WSO, our solution will also terminate in time polynomial in $n, T, \hat{\ell}, \log 1/\epsilon$ and the runtime of WSO on points of bit complexity $\text{poly}(n, T, \hat{\ell}, \log 1/\epsilon)$. Indeed, for every guess on the revenue and as long as the Ellipsoid algorithm for that guess has not terminated, it must be that WSO has been rejecting the points that it has been queried, and by Corollary 3.1 in this case it acts as a valid separation oracle for P_0, and hence is input points of the same bit complexity that could have been input to SO, namely $\text{poly}(n, T, \hat{\ell}, \log 1/\epsilon)$. Corollary 5.2 shows that the runtime of WSO on points of bit complexity $\text{poly}(n, T, \hat{\ell}, \log 1/\epsilon)$ is

$$\text{poly}(n, T, \hat{\ell}, 1/\epsilon, \log 1/\eta, 1/\gamma) \cdot rt_A(\text{poly}(n, T, \hat{\ell}, \log 1/\epsilon)),$$

so the entire running time of our algorithm is as promised. □

H Generic Theorem Statement

The following theorem summarizes our results for approximately optimizing over a convex polytope using a weird separation oracle. The summary is stated in a more generic form than is required for the main results of this paper so that it can be applied more easily in future work.

THEOREM H.1. *Let P be a d-dimensional bounded polytope containing the origin, and let $\mathcal{A}$ be any algorithm that takes any direction $\vec{w} \in [-1, 1]^d$ as input and outputs a point $\mathcal{A}(\vec{w}) \in P$ such that $\mathcal{A}(\vec{w}) \cdot \vec{w} \geq \alpha \cdot \max_{\vec{x} \in P} \vec{x} \cdot \vec{w}$ for some absolute constant $\alpha \in (0, 1]$. Then we can design a "weird" separation oracle WSO such that,*

1. *Every halfspace output by WSO will contain $\alpha P = \{\alpha \cdot \vec{\pi} \mid \vec{\pi} \in P\}$.*
2. *Whenever $WSO(\vec{x})$ = "yes," the execution of WSO explicitly finds directions $\vec{w}_1, \ldots, \vec{w}_k$ such that $\vec{x} \in Conv\{\mathcal{A}(\vec{w}_1), \ldots, \mathcal{A}(\vec{w}_k)\}$.*
3. *Let b be the bit complexity of $\vec{x}$, ℓ be an upper bound of the bit complexity of $\mathcal{A}(\vec{w})$ for all $\vec{w} \in [-1, 1]^d$, and $rt_{\mathcal{A}}(y)$ be the running time of algorithm $\mathcal{A}$ on some input with bit complexity y. Then on input $\vec{x}$, WSO terminates in time $poly(d, b, \ell, rt_{\mathcal{A}}(poly(d, b, \ell)))$ and makes at most $poly(d, b, \ell)$ many queries to $\mathcal{A}$.*

Proof. In fact, we have already proved all three claims in some previous Lemmas and Corollaries. For each claim, we point out where the proof is. For the first claim, using Corollary 3.1 we know P_1 is contained in all halfspaces output by WSO, therefore $\alpha P \subseteq P_1$ is also contained in all halfspaces. The second claim is proved in Lemma 3.2, and the third claim is proved in Corollary 5.2.

objective $\vec{w} \cdot \vec{x}$ over $\vec{x} \in F(\mathcal{F}, \mathcal{D}')$ to within a multiplicative factor $\alpha - \gamma$.

[14] Note that for any guess x on the revenue, we can upper bound the volume of the resulting polytope by $2^{O(T)}$ and lower bound it by some $2^{-\text{poly}(n, T, \hat{\ell}, \log 1/\epsilon)}$, whatever its dimension is. We can also take the precision to be $\text{poly}(n, T, \hat{\ell}, \log 1/\epsilon)$.