Mechanism Design: A New Algorithmic Framework

by

Yang Cai

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2013

Author .....
Department of Electrical Engineering and Computer Science
May 22, 2013

Certified by .....
Constantinos Daskalakis
Associate Professor of EECS
Thesis Supervisor

Accepted by .....
Leslie A. Kolodziejski
Chairman, Department Committee on Graduate Students

Mechanism Design: A New Algorithmic Framework

by

Yang Cai

Submitted to the Department of Electrical Engineering and Computer Science
on May 22, 2013, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Computer Science

Abstract

A modern engineering system, e.g. the Internet, faces challenges from both the strategic behavior of its self-interested participants and the inherent computational intractability of large systems. Responding to this challenge, a new field, Algorithmic Mechanism Design, has emerged. One of the most fundamental problems in this field is ***How to optimize revenue in an auction?***

In his seminal paper [Mye81], Myerson gives a partial solution to this problem by providing a revenue-optimal auction for a seller who is looking to sell a *single* item to multiple bidders. Extending this auction to simultaneously selling multiple heterogeneous items has been one of the central open problems in Mathematical Economics.

We provide such an extension that is also computationally efficient. Our solution proposes a novel framework for mechanism design by *reducing mechanism design problems* (where one optimizes an objective function on"*rational inputs*") *to algorithm design problems* (where one optimizes an objective function on"*honest inputs*"). Our reduction is generic and provides a framework for many other mechanism design problems.

Thesis Supervisor: Constantinos Daskalakis
Title: Associate Professor of EECS

Acknowledgments

First of all, I would like to thank my advisor, Costis Daskalakis, for all his guidance and encouragement. My whole Ph.D. has been a wonderful experience largely because of Costis. He has made himself available over these past four years whenever I want to have a discussion. I want to thank him for all the wise advices he has given to me on research, career and life. His enthusiasm and his taste in research problems have deeply influenced me and helped me to develop as a better researcher.

I also want to thank the theory faculty at MIT for the friendly, collaborative and stimulating atmosphere they have created for the group. In particular, I want to express my gratitude to Silvio Micali and Andrew Lo for serving on my dissertation committee and all their invaluable advices beyond this thesis.

I feel in debt to all my co-authors: Costis, Gagan Aggarwal, Zhiyi Huang, Fan Long, Haifeng Luo, Mohammad Mahdian, Aranyak Mehta, George Pierrakos, Bo Waggoner, Xi Wang, Matt Weinberg and Ting Zhang. Thanks for all the things they taught me! Special thanks to Matt for being my closest collaborator. I deeply enjoyed the days and nights we spent thinking, writing papers and rehearsing talks together! Without him, all the results in this thesis might be impossible.

I want to express my sincere thanks to Yuval Peres at Microsoft Research Redmond, as well as Gagan Aggarwal and Aranyak Mehta at Google Mountain View, for two very enjoyable summer internship. These experiences have greatly broadened my research horizon.

Thanks to all my friends at MIT for making the last five years wonderful: Pablo Azar, Haogang Chen, Jing Chen, Alan Deckelbaum, Shuo Deng, Mohsen Ghaffari, Bernhard Haeupler, Tao Lei, Huijia Lin, Fan Long, Yuan Luo, Yandong Mao, Krzysztof Onak, Yue Pan, Debmalya Panigrahi, Christoz Tzamos, Xi Wang, Matt Weinberg, Ning Xie, Morteza Zadimoghaddam, Zeyuan Zhu and everyone.

Last but not least, I am immensely grateful to my parents. I want to thank them the most for always being supportive for all my choices, and offering wise advices and selfless helps through my toughest time. I have no way to thank them for their

priceless love, care, inspiration, patience, and encouragement, but to dedicate this thesis to them.

Contents

List of Figures

Chapter 1

Introduction

Traditionally, engineering systems are centrally designed, and their various parts cooperate with each other to produce the desired outcome (think of the Integrated Circuit, for example). However, in modern engineering applications, systems have started to deviate from this paradigm. One of the most well-known examples is the Internet. It is not centrally designed, and its various components do not necessarily cooperate to produce a result but may *optimize their own strategic objectives*, resembling socio-economic behavior; e.g. a recent two-hour outage in global YouTube access resulted from a BGP table update due to censorship in Pakistan [Sto]. Because of such socio-economic phenomena, it is crucial to use concepts and ideas from Economics and Game Theory to understand the modern engineering systems.

The subfiled of Economics dedicated to the design and analysis of such complicated systems is called *Mechanism Design*. The goal is the design of optimal institutions and regulations even when the designer has limited information.

1.1 Algorithmic Mechanism Design

Mechanism Design: reverse game theory

Mechanism design is a unique branch of Game Theory and Economics: most of Game Theory and Economics analyzes existing economic systems, so that we can

predict what will happen in these systems or explain why some outcome has actually happened. In contrast, Mechanism design comes from a complete opposite direction. It starts with identifying some desired outcome, and then asks if it is possible to design a system such that when it is used by self-interested participants the desired outcome will arise. If yes, how?

The main difficulty for designing the right system is the information asymmetry between the designer and the participants: usually, how desirable an outcome is depends on the interests of the participants in some aggregated sense. However, the designer is usually ignorant about the participants. A participant's interest for some outcome is private information to the participant herself. Thus, it requires some cooperation of the participants to achieve a desirable outcome, e.g. honestly reporting how "valuable" an outcome is to each. Since the selected outcome depends on each participant's report, unless the system is designed carefully to properly *incentivize* a participant, e.g. promising a favorable outcome for her, monetary compensation or the combination of both, she may lie about her privately held information to manipulate the outcome to maximize her own objective at the expense of the whole system.

The Rise of Algorithmic Mechanism Design

Despite its difficulty, mechanism design has actually gained great success in both theory and practice. It addresses the economic side of our system design problem. However, in the context of the Internet, the number of participants is normally gigantic and growing. To guarantee the performance of a system, we have to take *computational efficiency* into consideration.

This is the mission of *Algorithmic Mechanism Design* (AMD), an area that aims to handle both the strategic behavior of self-interested participants and the inherent computational intractability of large systems. Primarily, AMD has roots in *Algorithm Design* from Computer Science, which focuses on optimizing an objective as a function of *" honest"* inputs with computational efficiency considerations. Inheriting the theoretical framework of *mechanism design*, AMD targets on optimizing an objective

as a function of "*rational*" inputs with computational efficiency considerations. The theory and concepts of AMD have been broadly and increasingly applied in the real world, from online market places (such as eBay) and sponsored search (Google, Bing) to spectrum auctions. For such applications, careful design might make a difference on the scale of billions of dollars. Thus, finding optimal mechanisms is a particularly meaningful topic not only in theory but also in the real world.

1.2 Fundamental Objectives

A mechanism design problem can be described by a set of feasible outcomes, and a group of agents, each equipped with a private valuation function mapping an outcome to a real number. The designer needs to choose an outcome that dependes on the agents' private valuation functions, and uses an objective function to "measures" the *quality* of the outcome.

There are two fundamental objectives in mechanism design:

- *Social Welfare* – the happiness of the agents participating in the mechanism
- *Revenue* – the happiness of the designer

Usually, researchers consider optimizing these objectives in the more concrete environment of multi-item auctions, because an auction is typically easier to describe yet has strong expressive power. See Section 2.2 for examples. In fact, any mechanism in which monetary transfers are allowed can be modeled as an auction.[1]

It is no surprise that the central problems in AMD are how to optimize these two fundamental objectives. To discuss existing results for these problems, we need to first explain how an auction is modeled.[2] Informally, there is an auctioneer/seller who has a limited supply of several items for sale, and there are many participants/bidders who are interested in these items. An auction is some communication protocol (e.g.

[1]Basically, for every outcome, we can use an item to represent it. Also, we use the following feasibility constraints on what allocations are legitimate: 1) every bidder can only receive one item; 2) every item can either be allocated to everyone or no one (similarly to a public object auction). From now on, we will use the word auction and mechanism interchangeably.

[2]Formal definitions can be found in Section 2.1.

each bidder reports her values for the items to the auctioneer) after which, the seller decides an allocation of the items and how much she wants to charge each bidder. If all the bidders are asked to do is to report their values to the auctioneer, and they are incentivized to do so honestly, the auction is called *truthful*.

Welfare Maximization

For social welfare, a beautiful mechanism was discovered in a sequence of works by Vickrey, Clarke and Groves [Vic61, Cla71, Gro73]. This mechanism is now known as the VCG mechanism, and it optimizes social welfare in all settings. This is an extremely general result and, surprisingly, the mechanism is very simple: all participants are asked to report how much they value each possible allocation[3]; then the allocation which maximizes the sum of all participants' values is chosen; in the end, the seller will carefully charge the participants to make sure the mechanism is truthful.

This completely solves the economic problem of designing welfare-maximizing mechanisms. When computational complexity is taken into account, finding the allocation which maximizes the sum of all participants value could be NP-hard. For computer scientists, the most natural way to overcome such barrier is to look for efficient approximations – a mechanism that approximately optimizes welfare. However, it turns out that this is a non-trivial task. The main reason is that if the end allocation is only approximately optimal, incentivizing the participants to tell the truth is very difficult. There has been a long line of research addressing this issue [NR99, LOS02, BKV05, DNS05, LS05, DNS06, BLP06, DN07, DD09, HL10, DR10, BH11, HKM11]. Recently in a sequence of works [HL10, BH11, HKM11], it is shown that in the Bayesian setting[4] if there is an algorithm for finding an approximately welfare optimal allocation, there is a generic way to turn it into a mechanism that optimizes welfare achieving the same approximation ratio. In other words, there is a *black-box*

[3]The values are not necessarily represented as a list. For example they could be modeled as an oracle that takes an allocation as input and outputs the value, if we care about computational efficiency.

[4]This is the standard setting in Mathematical Economics and is also the setting we use in this thesis. See Section 2.1 for a formal definition.

reduction from mechanism design to algorithm design for welfare optimization.

Revenue Maximization

Compared to welfare maximization, the progress for maximizing revenue has been much slower. The most important result is Myerson's Nobel-Prize winning work from 1981, which showed that if there is a *single* item for sale, a simple auction can achieve the optimal revenue [Mye81].

Myerson's Auction (informal)

1. Each bidder reports her value for the item.
2. The seller transforms each bidder's report to some "virtual"-report using some bidder specific function.
3. Apply the VCG allocation rule on the "virtual"-reports.

This elegant result is considered one of the milestones in mechanism design. A natural question raised by this result is, what if there are multiple items for sale? More specifically,

- **Are there simple, efficiently computable, revenue-optimal multi-item multi-bidder auctions?**

It is no surprise that this has become a central open problem in Mathematical Economics attracting lots of attention from not only that community but also the Theory of Computation community.

Despite continual effort of economists [MM88, Wil93, Arm96, RC98, Arm99, Zhe00, Bas01, Kaz01, Tha04, MV06, MV07], no major progress has occurred for over 30 years. Similarly, on the Computer Science side, although many works have designed constant factor approximations for optimizing revenue [CHK07, CHMS10, BGGM10, Ala11, HN12, KW12], all settings were heavily restricted, and no general solution had been provided.

One of the main reasons for the slow progress was that the right tool for this problem seemed to still be missing. In Myerson's result, the proof is purely algebraic and even seems magical. Many researchers have tried to understand and extend his approach to the general setting, but no one was able to do so. To solve the general case, a novel and drastically different approach is needed. Motivated by the importance of this problem, we devote this thesis to the problem of *revenue-optimal mechanism design*. Our result is not only a clear solution to this central open problem, but also provides a novel and general framework that can be applied to many other AMD problems.

1.3 Overview of Main Result and Techniques

Main result

We answer this open problem affirmatively under the technical assumption that the bidders are *additive*[5]: *there are simple, efficiently computable, revenue-optimal multi-item auctions, even in the most general settings.*

In particular, we provide a poly-time black box reduction from a mechanism design problem of finding a revenue-optimal auction, to an algorithm design problem of finding a welfare-maximizing allocation. More specifically, we reduce the problem to implementing the VCG allocation on some *"virtual"* reports instead of the real reports. The revenue-optimal auction we obtain looks like:

[5]See Section 2.1 for the formal definition. It basically means a bidder's value for a bundle of items equals to the sum of her value for each item.

The Revenue-optimal Auction (informal)

0. In a preprocessing step, the seller computes a distribution over virtual transformations, where each transformation has a bidder-specific transformation function for every bidder.
1. Each bidder reports her values for all items.
2. The seller samples one virtual transformation from the distribution computed in the preprocessing step.
3. The seller applies the sampled transformation to the real reports to get the *"virtual"* reports.
4. Use the VCG allocation rule to allocate the items based on the *"virtual"* reports.

So how does our solution compare to Myerson's single-item result? In Myerson's optimal auction, the allocation rule is just the VCG allocation rule, but on virtual reports instead of true reports. Then, he provides a closed-form description of the virtual transformations as well as a closed-form description of the pricing rule that makes the bidders report truthfully. However, in the general setting, it is known that randomness is necessary to achieve optimal revenue, even with a single bidder and two items [BCKW10, CMS10]. Hence, we cannot possibly hope for a solution as clean as Myerson's, but we have come quite close in a very general setting: Our allocation rule is just a distribution over virtual VCG allocation rules. Instead of a closed form for each virtual transformation and the pricing rule, we provide a computationally efficient algorithm to find them.

The Framework

Our approach also gives a new framework for solving a mechanism design problem, by reducing it to an algorithm design problem, that is, we reduce a problem with incentives to a problem without incentives. Our reduction can be divided into two

main steps: 1) Reduce the revenue-optimal mechanism design problem to the problem of checking feasibility of reduced form auctions. 2) Reduce the problem of checking feasibility of reduced form auctions to the problem of finding welfare-optimal allocations.

For the first step, our reduction heavily relies on linear programming and the ellipsoid method. In fact, several special cases of the revenue-optimal mechanism design problem have been solved computationally efficiently by linear programming [CH13, DW12]. Simply put, these algorithms explicitly store a variable for every possible bidder report profile denoting the probability that bidder i receives item j on that profile, and write a linear program to maximize expected revenue subject to the constraint that the auction must be feasible and the bidders must be truthful. Unfortunately, the number of variables required for such a program is exponential in the number of bidders, making such an explicit description prohibitive.

Our solution cleverly uses the *reduced form* of an auction to sidestep this dimensionality issue. The reduced form of an auction, first introduced in [Bor91, MR84, Mat84], is a succinct description of the auction. Intuitively, a reduced form auction can be viewed as promises that the seller makes to the bidders about the probabilities of receiving the items based on their own report. Using reduced form auctions as the variables of our LP has two major advantages: 1) its total size is polynomial in the number of bidders; 2) more importantly, it contains all the necessary information to verify that the bidders are truthful. That is, we can guarantee the bidders are honest by adding linear constraints to the LP.

The only obstacle for solving this LP is that we need to verify the feasibility of a reduced form auction efficiently. More specifically, given a reduced form auction, we need to verify if there is a feasible mechanism implementing it (i.e, whether it matches these probabilities), if yes, can we also construct this mechanism efficiently? This is the problem of checking feasibility of reduced form auctions. Hence, we successfully reduced the revenue-optimal mechanism design problem to this algorithmic problem.

It turns out checking feasibility of reduced form auctions is an interesting algorithmic problem by itself, and has already been studied by Border twenty years

ago [Bor91]. Unfortunately, Border only studied a special case when there is only a single item and he did not gave any efficient algorithm even for this special case. Our first contribution is to provide an efficient algorithm for this problem in Chapter 3. However, to solve the multi-item revenue-optimal mechanism design problem, we need to be able to check feasibility for general, i.e. multi-item, reduced form auctions. We show that for arbitrary feasibility constraints[6], we can construct an efficient separation oracle for checking feasibility of general reduced form auctions using only black-box calls to an algorithm that exactly maximizes welfare under the same constraints (Chapter 4). In other words, we have black-box reduced the problem of checking feasibility of reduced form auctions to the algorithmic problem of finding an allocation that exactly maximizes welfare. This completes the second step of our whole reduction.

However, this result requires the welfare-maximizing algorithm to be exact. For many feasibility constraints, this problem is intractable unless P=NP. In Chapter 6, we take one step further. We show that given any α approximation algorithm for maximizing welfare under some feasibility constraints, by making only black-box calls to this algorithm we can still construct an "approximate" separation oracle for checking feasibility. Using such an "approximate" separation oracle, we can design an auction that achieves at least α-fraction of the optimal revenue. Hence, our reduction can accommodate approximations.

In this thesis, we only study the objective of revenue, but clearly our framework can accommodate any objective that is concave in the reduced form auctions, e.g. social welfare or any convex combination of revenue and social welfare. In a very high level, our framework for mechanism design can be described as a two stage procedure. First, find a succinct representation of auction, which still contains all necessary information to verify if the bidders are truthful. Write an LP using this representation as the variables. This reduces the mechanism design problem to an algorithmic problem of checking feasibility for this succinct representation of auction. Second, design

[6]For example, bidders might be unit-demand, so an allocation is feasible iff it is a matching between the items and bidders . See Section 2.2 for a formal definition and more discussion.

an efficient algorithm to check feasibility. Our framework has already been used. In a recent unpublished manuscript [CDW13b], we introduced a new succinct description of auction, implicit forms, and extended our framework to accommodate new mechanism design problems, for example maximizing fractional max-min fairness.

1.4 Thesis Organization

In Chapter 2, we first introduce all mechanism design concepts and definitions that are needed to read this thesis. Next, we formally define the Revenue-Optimal Mechanism Design problem, and provide a more in-depth discussion of our main result, the black-box reduction from revenue to welfare. Then we overview the related work on the revenue-optimal mechanism design problem. In the end, we introduce some further notations and a few theorems and algorithms that will be repeatedly used in future chapters.

In Chapter 3, we study the feasibility of single-item reduced form auctions. Our result is enabled by a novel, constructive proof of Border's theorem [Bor91], and a new generalization of this theorem to independent (but not necessarily identically distributed) bidders, improving upon the results of [Bor07, CKM11]. For a single item and independent (but not necessarily identically distributed) bidders, we show that any feasible reduced form auction can be implemented as a distribution over hierarchical mechanisms. We also give a polynomial-time algorithm for determining feasibility of a reduced form auction, or providing a separation hyperplane from the set of feasible reduced forms. To complete the picture, we provide polynomial-time algorithms to find and exactly sample from a distribution over hierarchical mechanisms consistent with a given feasible reduced form.

In Chapter 4, we study the feasibility of general reduced forms auctions under a arbitrary feasibility constraint. We first show a characterization result that every feasible reduced form auction can be implemented as a distribution over *virtual VCG allocation rules*. A virtual VCG allocation rule has the following simple form: Every bidder's valuation t_i is transformed into a virtual valuation $f_i(t_i)$, via a bidder-specific

function. Then, the allocation maximizing virtual welfare is chosen. We generalize this result to arbitrarily correlated bidders, introducing the notion of a *second-order* VCG allocation rule. Next, we give two algorithmic results on reduced form auctions in settings with arbitrary feasibility and demand constraints. First, we provide a separation oracle for determining feasibility of a reduced form auction. Second, we provide a geometric algorithm to decompose any feasible reduced form into a distribution over virtual VCG allocation rules. In addition, we show how to efficiently execute both algorithms given only black box access to an implementation of the VCG allocation rule.

In Chapter 5, based on the separation oracle designed in Chapter 4, we provide a reduction from revenue maximization to welfare maximization in multi-item auctions with arbitrary (possibly combinatorial) feasibility constraints and independent bidders with arbitrary (possibly combinatorial) demand constraints, appropriately extending Myerson's result [Mye81] to the general setting.

In Chapter 6, we extend the results in Chapter 4 and 5 to accommodate approximations. In Chapter 5, we show that revenue optimization can be computationally efficiently reduced to welfare optimization in all multi-item auctions 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 chapter, 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 the earlier structural results on optimal mechanisms to approximately optimal mechanisms. Unlike in Chapter 4, our results are obtained through novel uses of the Ellipsoid algorithm and other optimization techniques over *non-convex regions*.

In Chapter 7, we give conclusions and open problems.

All results are based on joint work with Constantinos Daskalakis and S. Matthew Weinberg. Chapter 3 is based on [CDW12a], Chapter 4 and 5 are based on [CDW12b] and Chapter 6 is based on [CDW13a].

Chapter 2

Background

In this chapter, we first introduce all mechanism design concepts and definitions that are needed to read this thesis. Next, we formally define the Revenue-Optimal Mechanism Design problem, and provide a more in-depth discussion of our main result – the black-box reduction from (truthful) revenue optimization to (not necessary truthful) welfare optimization. Then we overview the related work on the revenue-optimal mechanism design problem. In the end, we introduce the notations and a few theorems and algorithms that will be repeatedly use in future chapters.

2.1 Basic Concepts from Mechanism Design

We provide the necessary mechanism design concepts in this section.

Basic Terms

Bidder's type: A bidder's type contains all information about the bidder that is relevant to the decision of the mechanism. For example, in a single item auction. A bidder's type will simply be her value for this item. Throughout this thesis, we will use t_i to denote bidder i's type and T_i be the set of possible types for bidder i.

Bayesian Setting: In a Bayesian setting, it is assumed that every bidder's type is drawn from a known distribution. By known, we mean the distribution is known to

the seller as well as the other bidders. Notice that only the distribution is known, not the exact value of the sample. Throughout this thesis, we will use $\mathcal{D}_i$ to denote bidder i's distribution, D_{-i} to denote the joint distribution of all bidders except i and $\mathcal{D}$ to denote the joint distribution of all bidders.

Utility Function: A utility function of a bidder is a function that maps an allocation, a payment and the bidder's type to a real value utility. We say the utility function is *quasi-linear* if it equals the bidder's value of the allocation minus her payment. We say a bidder is *risk-neutral* if her value for a distribution over allocations is her expected value for the sampled allocation.

(Direct Revelation) Mechanism: A mechanism is the following three-step protocol: (1) Each bidder submits her type, which is usually called her *bid*. (2) The seller use the bid profile to decide some allocation, and finally (3) The seller charges each bidder some monetary payment. The map from a bid profile to an allocation in step (2) is called an *allocation rule*, and the map from a bid profile to payments in step (3) is called a *pricing rule*.

Social Welfare and Revenue: Social welfare is the sum of every bidder's value for the allocation chosen by the mechanism. Revenue is the sum of every bidder's payment.

Essential Concepts

Truthfulness/Incentive compatibility: A mechanism is truthful/incentive compatible iff each bidder's utility is maximized when they report her type honestly. Formally, it is known as *Bayesian Incentive Compatible (BIC)*.

- *Bayesian Incentive Compatible (BIC):* Truthfully reporting maximizes a bidder's utility if other bidders are also truthfully reporting. That is, for any bidder i, her expected utility is maximized when being truthful, where the expectation is taken over the randomness of the mechanism and the randomness the other bidders' bids (assuming they are sampled from $\mathcal{D}_{-i}$).

Individual Rationality (IR): We say a mechanism is individual rational iff each bidder's utility is non-negative when they report her type truthfully. A mechanism is interim IR if each bidder's utility is non-negative when being truthful in expectation over other bidder's types. A mechanism is ex-post IR if each bidder's utility is non-negative when being truthful under any profile of bids.

Revelation Principle: A mechanism can actually have a more complicated communication protocol than a direct revelation one, which simply asks the bidders to submit their types. Such a mechanism is usually modeled as a *game of incomplete information*. However, the *Revelation Principle* states that for any outcome that can be achieved in an equilibrium of a game of incomplete information, there is a truthful direct revelation mechanism that achieves the same outcome and preserves every bidder's utility [Mye79]. In other words, it implies that for any mechanism design problem, focusing on only truthful direct revelation mechanism is WLOG.

2.2 The Optimal Mechanism Design Problem

The revenue-optimal mechanism design problem has received much attention from the Economics community, and recently the Computer Science community as well. The problem description is simple: a seller has a limited supply of several heterogenous items for sale and many interested buyers. The goal is for the seller to design an auction for the buyers to play that will maximize her revenue.

In order to make this problem tractable (not just computationally, but at all[1]), some assumptions must be made:

- First, we assume we are in a Bayesian setting.
- Second, the bidders will play any auction at a *Bayes-Nash Equilibrium*. We also assume that all buyers are *quasi-linear* and *risk-neutral*.

[1]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?

- Finally, we say that the goal of the seller is to maximize her *expected revenue* over all auctions when played at a Bayes-Nash Equilibrium.

All of these assumptions have become standard with regards to this problem. Indeed, all were made in Myerson's seminal paper on revenue-maximizing mechanism design where this problem is solved for a single item and product distributions [Mye81].

In addition, Myerson introduces the *revelation principle*, showing that every auction played at a Bayes-Nash Equilibrium is strategically equivalent to a *Bayesian Incentive Compatible* (BIC) *direct revelation* mechanism. In a direct revelation mechanism, each bidder reports a bid for each possible subset of items they may receive. In essence, Myerson's revelation principle says that one only needs to consider BIC direct revelation mechanisms rather than arbitrary auctions played at a Bayes-Nash Equilibrium to maximize revenue (or any other objective for that matter).

In Myerson's result, a single real number suffices to represent a bidder's type. However, in multi-item auctions, we need multiple numbers to specify a bidder's type. For example, when a bidder's valuation is additive, we need one number for each item. That is why the Revenue-Optimal Mechanism Design problem is usually called the *multi-dimensional mechanism design problem*.

As we depart from Myerson's single-item setting, the issue of *feasibility* arises. With only a single item for sale, it is clear that the right feasibility constraints are simply that the item is always awarded to at most a single bidder, but it will be much more complicated in multi-item cases. Imagine natural scenarios with heterogenous items:

- Maybe the items are houses. In this case, a feasible allocation awards each house to at most one bidder, and to each bidder at most one house.

- Maybe the items are appointment slots with doctors. In this case, a feasible allocation does not award the same slot to more than one bidder, and does not award a bidder more than one slot with the same doctor, or overlapping slots with different doctors.

- Maybe the items are bridges built at different locations. In this case, if a bridge is built, everyone will be able to use it, so a feasible allocation awards each bridge to everyone or to no one.

  ...

Just like in the above cases, feasibility constraints can be all different. Sometimes, feasibility constraints are imposed by the supply side of the problem: a doctor cannot meet with two patients at once, and a bridge cannot be built for one bidder but not another. Other times, feasibility constraints are imposed by the demand side of the problem: no bidder wants two houses or two appointments with the same doctor.

Without differentiating where feasibility constraints come from, we model them in the following way: let $\mathcal{A} = [m] \times [n]$ denote the space of assignments (where (i, j) denotes that bidder i is assigned item j), and let $\mathcal{F}$ be a set system on $\mathcal{A}$ (that is, a subset of $2^{\mathcal{A}}$). Then in a setting with feasibility constraints $\mathcal{F}$, it is possible for the seller to simultaneously make any subset of assignments in $\mathcal{F}$. $\mathcal{F}$ may be a truly arbitrary set system, *it need not even be downward-closed*.[2]

As we leave the single-item setting, we also need to consider how a bidder values a bundle of multiple items. In general, a bidder may have arbitrarily complicated ways of evaluating bundles of items, and this information is encoded into the bidder's type. For the problem to be computationally meaningful, however, one would want to either assume that the auctioneer only has oracle access to a bidder's valuation, or impose some structure on the bidders' valuations allowing them to be succinctly described. Indeed, virtually every recent result in revenue-maximizing literature [AFH$^+$12, BGGM10, CD11, CH13, CHK07, CHMS10, DW12, KW12] assumes that bidders are *capacitated-additive*.[3] In fact, most results are for unit-demand bidders. It is easy to see that, if we are allowed to incorporate arbitrary demand constraints into the definition of $\mathcal{F}$, such bidders can be described in our model as simply additive. In fact, far more complex bidders can be modeled as well, as demand con-

[2] We say a set system $\mathcal{F}$ is downward-closed, if for any set $S \in \mathcal{F}$ all subsets of S are also in $\mathcal{F}$.

[3] A bidder is capacitated-additive if for some constant C her value for any subset S of at most C goods is equal to the sum of her values for each item in S, and her value for any subset S of more than C goods is equal to her value for her favorite $S' \subseteq S$ of at most C goods.

straints could instead be some arbitrary set system. Because $\mathcal{F}$ is already an arbitrary set system, we may model bidders as simply additive and still capture virtually every bidder model studied in recent results, and more general ones as well. In fact, we note that every multi-dimensional setting can be mapped to an additive one, albeit not necessarily computationally efficiently.[4] So while we focus our discussion to additive bidders throughout this thesis, our results apply to every auction setting, without need for any additivity assumption. In particular, our characterization result (Informal Theorem 2) of feasible allocation rules holds for any multi-dimensional setting, and our reduction from revenue to welfare optimization (Informal Theorem 1) also holds for any setting, and we show that it can be carried out computationally efficiently for any additive setting. In Section 6.9, we generalize our result to settings beyond additive and introduce a new concept called "additive dimension". We show that if a setting has "additive dimension" d, all results for additive setting still hold after multiplying the runtime by only a poly(d) factor.

Optimal Multi-dimensional Mechanism Design. With the above motivation in mind, we formally state the revenue optimization problem we solve. We remark that virtually every known result in the multi-dimensional mechanism design literature (see references above) tackles a special case of this problem, possibly with budget constraints on the bidders (which can be easily incorporated in all results presented in this thesis as discussed in Section 5.3), and possibly replacing BIC with DSIC.[5] We explicitly assume in the definition of the problem that the bidders are additive, recalling that this is not a restriction if computational considerations are not in place.

[4]The generic transformation is to introduce a meta-item for every possible subset of the items, and have the feasibility constraints (which are allowed to be arbitrary) be such that an allocation is feasible if and only if each bidder receives at most one meta-item, and the corresponding allocation of real items (via replacing each meta-item with the subset of real items it represents) is feasible in the original setting. Many non-additive settings allow much more computationally efficient transformations than the generic one.

[5]Dominant Strategy Incentive Compatible (DSIC) is another truthfulness definition, and is usually seen in the literature of welfare maximization. In particular, DSIC means truthfully bidding maximizes a bidder's utility no matter what the other bidders report.

Revenue-Maximizing Multi-Dimensional Mechanism Design Problem (MDMDP): Given as input m distributions (possibly correlated across items) $\mathcal{D}_1, \ldots, \mathcal{D}_m$ over valuation vectors for n heterogenous items and feasibility constraints $\mathcal{F}$, output a BIC 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 whose valuation vectors are sampled from $\mathcal{D} = \times_i \mathcal{D}_i$.

2.3 Black-box Reduction from Revenue to Welfare

We provide a poly-time black box reduction from the MDMDP with feasibility constraints $\mathcal{F}$ to implementing VCG with feasibility constraints $\mathcal{F}$ by introducing the notion of a *virtual VCG allocation rule*. A virtual VCG allocation rule is defined by a collection of functions f_i for each bidder i. f_i takes as input bidder i's reported bid vector and outputs a virtual bid vector. When the reported bids are $t_1, \ldots, t_m$, the virtual VCG allocation rule with functions $\{f_i\}_{i \in [m]}$ simply implements the VCG allocation rule (with feasibility constraints $\mathcal{F}$) on the virtual bid vectors $f_1(t_1), \ldots, f_m(t_m)$. We also note here that implementing VCG for additive bidders is in general *much* easier than implementing VCG for arbitrary bidders.[6] Our solution to the MDMDP is informally stated below, and is formally given as Theorem 17 of Section 5.1:

Informal Theorem 1. *Let $A_{\mathcal{F}}$ be an implementation of the VCG allocation rule with respect to $\mathcal{F}$ (i.e. $A_{\mathcal{F}}$ takes as input a profile of bid vectors and outputs the VCG allocation). Then for all $\mathcal{D}_1, \ldots, \mathcal{D}_m$ with finite support and all $\mathcal{F}$, given $\mathcal{D}_1, \ldots, \mathcal{D}_m$ and black box access to $A_{\mathcal{F}}$ (and without need of knowledge of $\mathcal{F}$), there exists a fully polynomial-time randomized approximation scheme[7] for the MDMDP whose runtime is polynomial in n, the number of bidder types (and* not *type profiles), and the runtime of $A_{\mathcal{F}}$. Furthermore, the allocation rule of the output mechanism is a distribution over*

[6] When bidders are additive, implementing VCG is simply solving the following problem, which is very well understood for a large class of feasibility constraints: every element of $\mathcal{A}$ has a weight. The weight of any subset of $\mathcal{A}$ is equal to the sum of the weights of its elements. Find the max-weight subset of $\mathcal{A}$ that is in $\mathcal{F}$.

[7] This is often abbreviated as FPRAS, and we provide its formal definition in Section 2.5.

virtual VCG allocation rules.

We remark that the functions defining a virtual VCG allocation rule may map a bidder type to a vector with negative coordinates. Therefore, our given implementation of the VCG allocation rule should be able to handle negative weights. This is not a restriction for arbitrary downwards-closed $\mathcal{F}$ as any implementation of VCG that works for non-negative weights can easily be (in a black-box way) converted into an implementation of VCG allowing arbitary (possibly negative) inputs.[8] But this is not necessarily true for non downwards-closed $\mathcal{F}$'s. If the given $A_{\mathcal{F}}$ cannot accommodate negative weights, we need to replace it with an algorithm that can in order for our results to be applicable.

Several extensions are stated and discussed in Section 5.1, including solutions for distributions of infinite support, and improved runtimes in certain cases that make use of techniques from [DW12]. We also extend all our solutions to accommodate strong budget constraints by the bidders in Section 5.3.

We further generalize Informal Theorem 1 in Chapter 6. Informal Theorem 1 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 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 Chapter 6 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.

Characterization of Feasible Reduced Form Auctions. In addition to our solution of the MDMDP, we provide a characterization of feasible reduced forms of multi-dimensional mechanisms in all (not necessarily additive) settings.[9] We show the

[8]The following simple black-box transformation achieves this: first zero-out all negative coordinates in the input vectors; then run VCG; in the VCG allocation, 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.

[9]For non-additive settings, the characterization is more usable for the purposes of mechanism design when applied to meta-items (see discussion above), although it still holds when directly applied to items as well.

following informal theorem, which is stated formally as Theorem 10 in Section 4.1. Recall that a virtual VCG allocation rule is associated with a collection of functions f_i that map types t_i to virtual types $f_i(t_i)$ for each bidder i, and allocates the items as follows: for a given type vector $(t_1, ..., t_m)$, the bidders' types are transformed into virtual types $(f_1(t_1), \ldots, f_m(t_m))$; then the virtual welfare optimizing allocation is chosen.

Informal Theorem 2. *Let $\mathcal{F}$ be any set system of feasibility constraints, and $\mathcal{D}$ any (possibly correlated) distribution over bidder types. Then the reduced form of any feasible mechanism can be implemented as a distribution over virtual VCG allocation rules.*

2.4 Related Work

2.4.1 Structural Results

Some structural results are already known for special cases of the MDMDP and its extension to correlated bidders. As we have already discussed, Myerson showed that the revenue-optimal auction for selling a single item is a virtual Vickrey auction: bids are transformed to virtual bids, and the item is awarded to the bidder with the highest non-negative virtual value [Mye81]. It was later shown that this approach also applies to all single-dimensional settings (i.e. when bidders can't tell the difference between different houses, appointment slots, bridges, etc) as long as bidders' values are independent. In this setting, bids are transformed to virtual bids (via Myerson's transformation), and the virtual-welfare-maximizing feasible allocation is chosen. These structural results are indeed strong, but hold only in the single-dimensional setting and are therefore of very limited applicability.

On the multi-dimensional front, it was recently shown that similar structure exists in restricted settings. It is shown in [CDW12a] that when selling multiple heterogenous items to additive bidders with *no* demand constraints (i.e. $\mathcal{F}$ only ensures that each item is awarded to at most one bidder), the optimal auction *randomly* maps bids

to virtual bids (according to some function that depends on the distributions from which bidders' values are drawn), then *separately* allocates each item to the highest virtual bidder.[10] It is shown in [AFH+12] that when there are many copies of the same customizable item and a matroid constraint on which bidders can simultaneously receive an item (i.e. $\mathcal{F}$ only ensures that at most k items are awarded, subject to a matroid constraint on the served bidders), that the optimal auction randomly maps bids to virtual bids (according to some function that depends on the distributions from which bidders' values are drawn), then allocates the items to maximize virtual surplus (and customizes them after). We emphasize that both results, while quite strong for their corresponding settings, are extemely limited in the settings where they can be applied. In particular, neither says anything about the simple setting of selling houses to unit-demand bidders (i.e. $\mathcal{F}$ ensures that each house is awarded at most once and each bidder receives at most one house: Example 1, Section 2.2). Selling houses to unit-demand bidders is on the easy side of the settings considered in this thesis, as we provide a solution in multi-dimensional settings with arbitrary feasibility constraints. We do not even assume that $\mathcal{F}$ is downward-closed.

For correlated bidders, the series of results by Cremer and McLean [CM85, CM88] and McAfee and Reny [MR92] solve for arbitrary feasibility constraints subject to a non-degeneracy condition on the bidder correlation (that is not met when bidders are independent). Under this assumption, they show that the optimal auction extracts full surplus (i.e. has expected revenue equal to expected welfare) and simply uses the VCG allocation rule (the prices charged are not the VCG prices, but a specially designed pricing menu based on the bidder correlation). Our structural results for correlated bidders apply to *arbitrary* feasibility constraints as well as *arbitrary* bidder correlation, removing the non-degeneracy assumption. Of course, the expected revenue extracted by our mechanisms cannot possibly always be as high as the expected maximum social welfare (as it happens in Cremer-McLean and McAfee-Reny) as they

[10] In fact, the allocation rule of [CDW12a] has even stronger structure in that each item is independently allocated to the bidder whose virtual value for that item is the highest, and moreover the random mapping defining virtual values for each item simply irons a *total ordering* of all bidder types that depends on the underlying distribution. This result is also included in Chapter 3.

also apply to independent bidders, but our characterization is still quite simple: the optimal auction randomly maps pairs of actual bids and possible alternative bids to second-order bids. Then, the second-order bids are combined (based on the underlying bidder correlation) to form virtual bids, and the virtual-welfare-maximizing allocation is chosen.

2.4.2 Algorithmic Results

The computer science community has contributed computationally efficient solutions to special cases of the MDMDP in recent years. Many are constant factor approximations [Ala11, BGGM10, CHK07, CHMS10, KW12]. These results cover settings where the bidders are unit-demand (or capacitated-additive) and the seller has matroid or matroid-intersection constraints on which bidders can simultaneously receive which items. All these settings are special cases of the MDMDP framework solved in this thesis.[11] In even more restricted cases near-optimal solutions have already been provided. Tools are developed in [CD11, CH13, DW12] that yield solutions for simple cases with one or few bidders. Cases with many asymmetric independent bidders is considered in [AFH$^+$12]. They studied the case where $\mathcal{F}$ ensures that at most k items are awarded, subject to a matroid constraint on the served bidders is solved. Our computational results push far beyond existing results, providing a computationally efficient solution in multi-dimensional settings with arbitrary feasibility constraints.

2.4.3 Black-box Reduction in Mechanism Design

One appealing feature of our result is that our reduction from approximate revenue optimization to non-truthful welfare approximation is black-box. 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 [BKV05, BLP06, HL10, DR10,

[11] Again, in some of these results [Ala11, BGGM10] bidders may also have budget constraints, which can be easily incorporated to the MDMDP framework without any loss, as is shown in Section 5.3, and some replace BIC with DSIC [Ala11, CHK07, CHMS10, KW12].

BH11, HKM11]. 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.

2.5 Preliminaries and notation

We denote the number of bidders by m, the number of items by n. To ease notation, we sometimes use A (B, C, etc.) to denote the *type* of a bidder, without emphasizing whether it is a vector or a scalar. 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 distribution $\mathcal{D}$ over $\times_i T_i$. For independent bidders, we use $\mathcal{D}_{-i}$ to denote the marginal of $\mathcal{D}$ over the types of all bidders, except bidder i. For correlated bidders, we use $\mathcal{D}_{-i}(\vec{v}_i)$ to denote the conditional distribution over the types of all bidders except for i, conditioned on bidder i's type being $\vec{v}_i$. We use t_i for the random variable representing the type of bidder i. So when we write $\Pr[t_i = A]$, we mean the probability that bidder i's type is A. In Section 2.7, we discuss how our algorithms access distribution $\mathcal{D}$.

We let $\mathcal{A} = [m] \times [n]$ 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 $\mathcal{A}$ (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) *mechanism*. As discussed in Section 2.2, we may also have a set system $\mathcal{F}$ on $\mathcal{A}$ (that is, a subset of $2^{\mathcal{A}}$), encoding constraints on what assignments can be made simultaneously by the mechanism. $\mathcal{F}$ 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 $\mathcal{F}$.

The *reduced form* of an allocation rule (also called the *interim allocation rule*) is a vector function $\pi(\cdot)$, specifying values $\pi_{ij}(A)$, for all items j, bidders i and types

$A \in T_i$. $\pi_{ij}(A)$ is the probability that bidder i receives item j when truthfully reporting type A, where the probability is over the randomness of all other bidders' types (drawn from $\mathcal{D}_{-i}$ in the case of independent bidders, and $\mathcal{D}_{-i}(A)$ in the case of correlated bidders) and the internal randomness of the allocation rule, assuming that the other bidders report truthfully their types. For single-item reduced forms, we omit the subscript j for convenience. When bidders are i.i.d., we say a reduced form is *bidder-symmetric* if $\pi_i(A) = \pi_{i'}(A)$ for all $i, i' \in [n]$ and any type A. 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 $M_{ij}(A)$ that bidder i receives item j when truthfully reporting type A is exactly $\pi_{ij}(A)$, where the probability is computed with respect to the randomness in the allocation rule and the randomness in the types of the other bidders, assuming that the other bidders report truthfully. While viewing reduced forms as vectors, we will denote by $F(\mathcal{F}, \mathcal{D})$ the set of feasible reduced forms when the feasibility constraints are $\mathcal{F}$ and consumers 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. If bidders are additive, to specify the preferences of 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 demand constraints, the *value* of additive bidder i of type $\vec{v}_i$ for a randomized allocation 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 $\sum_j v_{ij} \cdot \pi_{ij} - p_i$.

Throughout this thesis, we denote by OPT the expected revenue of an optimal solution to MDMDP. Also, most of our results for this problem construct a *fully polynomial-time randomized approximation scheme*, or FPRAS. This is an algorithm

that takes as input two additional parameters $\epsilon, \eta > 0$ and outputs a mechanism (or succinct description thereof) whose revenue is at least $\text{OPT} - \epsilon$, with probability at least $1-\eta$ (over the coin tosses of the algorithm), in time polynomial in $n \sum_i |T_i|, 1/\epsilon$, and $\log(1/\eta)$.

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 $\mathcal{D}_1, \ldots, \mathcal{D}_m$.

Also, for completeness, we formally define in Section 2.6 the standard notion of Bayesian Incentive Compatibility (BIC) and Individual Rationality (IR) of mechanisms for independent bidders, and state a well-known property of the Ellipsoid Algorithm for linear programs.

2.6 Details from Preliminaries

We provide a formal definition of Bayesian Incentive Compatibility and Individual Rationality of a mechanism for independent bidders and state a well-known property of the Ellipsoid Algorithm. For completeness, we provide an additional proposition showing a standard trick that can force the Ellipsoid algorithm to always output a corner.

Definition 1. *[DW12](BIC/ϵ-BIC Mechanism) A mechanism M is called ϵ-BIC iff the following inequality holds for all bidders i and types $\tau_i, \tau_i' \in T_i$:*

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

where:

- $U_i(A, M_i(B\ ;\ t_{-i}))$ *denotes the utility of bidder* i *for the outcome of mechanism* M *if his true type is* A*, he reports* B *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*

- $\pi_{ij}^M(A)$ *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 report truthfully, and 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* τ_i' *receives. A mechanism is called BIC iff it is* 0*-BIC.*[12]

We also define individual rationality of BIC/ϵ-BIC mechanisms:

Definition 2. *A BIC/*ϵ*-BIC mechanism* M *is called* interim individually rational (interim IR) *iff for all bidders* i *and types* $\tau_i \in T_i$*:*

$$\mathbb{E}_{t_{-i}\sim\mathcal{D}_{-i}}\left[U_i(\tau_i, M_i(\tau_i\ ;\ t_{-i}))\right] \geq 0,$$

where $U_i(A, M_i(B\ ;\ t_{-i}))$ *denotes the utility of bidder* i *for the outcome of mechanism* M *if his true type is* A*, he reports* B *to the mechanism, and the other bidders report* t_{-i}*. The mechanism is called* ex-post individually rational (ex-post IR) *iff for all* i*,* τ_i *and* t_{-i}*,* $U_i(\tau_i, M_i(\tau_i\ ;\ t_{-i}) \geq 0$ *with probability* 1 *(over the randomness in the mechanism).*

[12]Strictly speaking, the definition of BIC in [DW12] 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 [DW12] is certainly considered ϵ-BIC by this definition. We basically call a mechanism ϵ-BIC if either the definition in [BH11, HKM11, HL10] ($\epsilon v_{\max}$) or [DW12] ($\epsilon v_{\max} \sum_j \pi_{ij}(\vec{w}_i)$) holds.

Theorem 1. *[Ellipsoid Algorithm for Linear Programming] Let P be a convex polytope in $\mathbb{R}^d$ specified via a separation oracle SO, and $\vec{c}\cdot\vec{x}$ be a linear function. Assume that all coordinates of $\vec{a}$ and b, for all separation hyperplanes $\vec{a}\cdot\vec{x}\leq b$ possibly output by SO, and all coordinates of $\vec{c}$ are rational numbers of bit complexity ℓ. Then we can run the ellipsoid algorithm to optimize $\vec{c}\cdot\vec{x}$ over P, maintaining the following properties:*

1. *The algorithm will only query SO on rational points with bit complexity poly(d,ℓ).*
2. *The ellipsoid algorithm will solve the Linear Program in time polynomial in d, ℓ and the runtime of SO when the input query is a rational point of bit complexity poly(d,ℓ).*
3. *The output optimal solution is a corner of P.*[13]

Proposition 1. *Let $\vec{a}$ be a d-dimensional vector, whose coordinates are rational numbers of bit complexity ℓ_1, P be a d-dimensional convex polytope, in which all coordinates of all corners are rational numbers of bit complexity ℓ_2. Then we can transform $\vec{a}$ into a new d-dimensional vector $\vec{b}$, whose coordinates are all rational numbers of bit complexity $d(\ell_1+1)+(2d^2+1)\ell_2+1$, such that $\vec{x}^*=\text{argmax}_{\vec{x}\in P}\,\vec{b}\cdot\vec{x}$ is unique. Furthermore,*

Proof. Let $a_i=p_i/q_i$, where both p_i and q_i are integers with at most ℓ_1 bits. Now change the a_i's to have the same denominator $Q=\Pi_i q_i$. So $a_i=p_i'/Q$, where $p_i'=p_i\Pi_{j\neq i}q_j$. Both Q and p_i' have at most $d\ell_1$ bits. Let now $b_i=(p_i'+2^{-(1+\ell_2+(2d\ell_2+1)\cdot i)})/Q$. So b_i can be described with $d(\ell_1+1)+(2d^2+1)\ell_2+1$ bits.

Now we will argue that, for any rational vector $\vec{z}\neq\vec{0}$, whose coordinates can be described with at most $2\ell_2$ bits, $\vec{b}\cdot\vec{z}\neq 0$. Let $z_i=r_i/s_i$, where both r_i and s_i are integers with at most $2\ell_2$ bits. Now modify z_i to be r_i'/S, where $S=\Pi_i s_i$ and $r_i'=r_i\Pi_{j\neq i}s_i$. Both S and r_i' have at most $2d\ell_2$ bits. Now consider the fractional

[13] This is well-known, but also proved in Proposition 1 for completeness.

parts of $Q \cdot S \cdot (\vec{b} \cdot \vec{z})$, which is

$$\sum_{i=1}^{d} 2^{-(1+\ell_2+(2d\ell_2+1)\cdot i)} \cdot r'_i.$$

But this equals to 0 only when $r_i = 0$ for all i. Thus, if $\vec{z} \neq \vec{0}$, $\vec{b} \cdot \vec{z} \neq 0$.

Next, we argue that, if $\vec{x}$ and $\vec{y}$ are two different vectors, whose coordinates can be described with ℓ_2 bits, $\vec{b} \cdot \vec{x} \neq \vec{b} \cdot \vec{y}$. This is implied by the above argument, since all coordinates of $\vec{x} - \vec{y}$ can be described with at most $2\ell_2$ bits. So there is a unique optimal solution to $\max_{\vec{x} \in P} \vec{b} \cdot \vec{x}$. Call that solution $\vec{x}^*$.

Now we show that $\vec{x}^*$ is also an optimal solution for $\max_{\vec{x} \in P} \vec{a} \cdot \vec{x}$. We only need to argue that if corner $\vec{x}$ is not optimal for $\vec{a}$, it will not be optimal for $\vec{b}$. First, it is not hard to see that for corners $\vec{x}$ and $\vec{y}$, if $\vec{a} \cdot (\vec{x} - \vec{y}) \neq 0$, $\vec{a} \cdot (\vec{x} - \vec{y}) \geq \frac{1}{2^{2d\ell_2} Q}$. Second, for any corner $\vec{x}$,

$$|(\vec{b} - \vec{a}) \cdot \vec{x}| \leq \sum_{i=1}^{d} \left| \frac{2^{-(1+\ell_2+(2d\ell_2+1)\cdot i)}}{Q} \right| \cdot 2^{\ell_2} < \frac{1}{2^{1+2d\ell_2} Q}.$$

So if $\vec{a} \cdot \vec{x} > \vec{a} \cdot \vec{y}$, $\vec{b} \cdot \vec{x}$ is still strictly greater than $\vec{b} \cdot \vec{y}$. Thus, $\vec{x}^*$ must be an optimal solution for $\max_{\vec{x} \in P} \vec{a} \cdot \vec{x}$. □

2.7 Input Model

We discuss two models for accessing a value distribution $\mathcal{D}$, 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 this thesis 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 [DW12] is a sketch of the modifications necessary for all our results to apply with 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.

2.8 A Geometric Algorithm

Carathéodory's theorem states that every point $\vec{x}$ inside an n-dimensional polytope P can be written as a convex combination of at most $n+1$ corners of P. In this section, we provide an efficient algorithm for coming up with such a combination.[14] We will consider polytopes that are described as an intersection of half-spaces. Each half-space is defined by a hyperplane h together with a choice of a side. We use $B(P)$ to denote the set of half-spaces, but overload notation using $B(P)$ to also denote the set of boundary hyperplanes of the polytope P. We reserve the symbol h to denote hyperplanes. In addition, we consider cases where $|B(P)|$ may be exponentially large, and we only have an implicit description of $B(P)$. That is, we have access to a *boundary oracle* BO that outputs yes on input h if $h \in B(P)$, and no otherwise. We also have access to a separation oracle, SO, that outputs yes on input $\vec{x}$ if $\vec{x} \in P$, and outputs some $h \in B(P)$ if $\vec{x}$ is on the wrong side of h (and therefore not in P). We will talk about one more algorithm related to P:

Definition 3. *CO is a **corner oracle** for P if it has the following behavior. Given as input a set of hyperplanes B, CO outputs no if $B \not\subseteq B(P)$, or $\left(\bigcap_{h\in B} h\right) \bigcap P = \emptyset$ (i.e. the hyperplanes are not boundary hyperplanes of P, or they are boundary hyperplanes, but do not intersect inside P). Otherwise, CO outputs a corner of P inside $\bigcap_{h\in B} h$.*

[14] Such an algorithm is in fact quite standard, given a separation oracle. We include it just for completeness.

It is clear that CO has well-defined behavior on all inputs. If B contains only boundary hyperplanes of P, and the intersection of these hyperplanes with P is non-empty, this region must contain a corner of P. Now we describe our algorithmic problem in terms of these algorithms:

Question 1. *Given as input a boundary oracle, BO, separation oracle SO, and corner oracle CO for some n-dimensional polytope P, and a point $\vec{x}$, output no if $\vec{x} \notin P$. Otherwise, output $c_1, \ldots, c_{n+1}$, $\vec{a}_1, \ldots, \vec{a}_{n+1}$ such that $\vec{a}_i$ is a corner of P for all i, $\sum_i c_i = 1$, and $\sum_i c_i \vec{a}_i = \vec{x}$.*

It follows from Carathéodory's theorem that such $c_i, \vec{a}_i$ exist whenever $\vec{x} \in P$. We provide an algorithm to find such a solution whenever it exists. At a high level, we begin with the input $\vec{x}$ and maintain at all times two points $\vec{y} \in P$, $\vec{z} \in P$, such that $\vec{x} = c\vec{y} + (1-c)\vec{z}$, for some $c \in [0,1]$. After step t of the algorithm is completed, $\vec{y}$ is the convex combination of at most t corners of P, and $\vec{z}$ lies in the $(n-t)$-dimensional intersection of t hyperplanes of $B(P)$. Hence, after at most n steps, $\vec{z}$ will lie in a 0-dimensional space, and therefore must be a corner, so the algorithm will terminate after at most $n+1$ steps.

To go from step t to step $t+1$, we pick an arbitrary corner, $\vec{a}_t$, that lies in the intersection of the t hyperplanes where $\vec{z}$ lies. Then, we let c_t be as large as possible without pushing the point $\frac{(1-\sum_{j<t} c_j)\vec{z} - c_t \cdot \vec{a}_t}{1-c_t-\sum_{j<t} c_j}$ outside of P. We update $\vec{z}$ to $\vec{z}_{\text{new}} = \frac{(1-\sum_{j<t} c_j)\vec{z}_{\text{old}} - c_t \cdot \vec{a}_t}{1-c_t-\sum_{j<t} c_j}$ and update $\vec{y}$ appropriately to include $\vec{a}_t$ in its convex combination of corners. The new $\vec{z}$ must lie at the intersection of the original t hyperplanes where the old $\vec{z}$ lied, as well as a new $h \in B(P)$ that stopped us from further increasing c_t. Below is a formal description of the algorithm. In the description, E denotes the set of hyperplanes whose intersection contains $\vec{z}$ (the empty intersection is the entire space).

Theorem 2. *Let P be a n-dimensional polytope with corner oracle CO and separation oracle SO such that each coefficient of every hyperplane ever output by SO is a rational number of bit complexity b. Then Algorithm 1 decomposes any point $\vec{x} \in P$ into a convex combination of at most $n+1$ corners of P. Furthermore, if ℓ is the*

Algorithm 1 Algorithm for writing $\vec{x}$ as a convex combination of at most $n+1$ corners

1: Initialize: $i := 1$, $\vec{y} := \vec{0}$, $\vec{z} := \vec{x}$, $E := \emptyset$, $c_i := 0, \vec{a}_i := \vec{0}\ \forall i \in [n+1]$.
2: Invariants: $c := \sum_i c_i$, $\vec{y} := \frac{1}{c}\sum_i c_i\vec{a}_i$, or $\vec{0}$ if $c = 0$, $c\vec{y} + (1-c)\vec{z} = \vec{x}$.
3: **if** $SO(\vec{x}) \neq$ yes **then**
4: Output no.
5: **end if**
6: **while** $c < 1$ **do**
7: Set $\vec{a}_i := CO(E)$.
8: **if** $\vec{a}_i = \vec{z}$ **then**
9: Set $c_i := 1 - c$.
10: Output $c_1, \ldots, c_{n+1}$, $\vec{a}_1, \ldots, \vec{a}_{n+1}$.
11: **else**
12: Set $D := \max\{d \mid (1+d)\vec{z} - d\vec{a}_i \in P\}$.
13: Set $E_i = SO((1+D+\epsilon)\vec{z} - (D+\epsilon)\vec{a}_i)$ for sufficiently small $\epsilon > 0$. */* the appropriate choice of ϵ is explained in the proof of Theorem 2*/*
14: Update: $c_i := (1 - \frac{1}{1+D})(1-c)$, $\vec{z} := \frac{1-c}{1-c-c_i}\vec{z} - \frac{c_i}{1-c-c_i}\vec{a}_i$, $\vec{y} := \frac{c}{c+c_i}\vec{y} + \frac{c_i}{c+c_i}\vec{a}_i$, $c := c + c_i$, $E := E \cup E_i$, $i := i+1$.
15: **end if**
16: **end while**

maximum number of bits needed to represent a coordinate of $\vec{x}$, then the runtime is polynomial in d, b, ℓ and the runtimes of SO and CO on inputs of bit complexity poly(n, b, ℓ).

Proof. First, we describe how to execute Steps 12 and 13 of the algorithm, as it is clear how to execute every other step. Step 12 can be done by solving a linear program using SO. Explicitly, maximize d subject to $(1+d)\vec{z} - d\vec{a}_i \in P$. For Step 13, we will explain later in the proof how to choose an ϵ small enough so that the following property is satisfied:

(P): for all $h \in B(P)$ and for whatever D is computed in Step 12 of the algorithm, if $(1+D)\vec{z} - D\vec{a}_i$ is not contained in h, then $(1+D+\epsilon)\vec{z} - (D+\epsilon)\vec{a}_i$ is on the same side of h as $(1+D)\vec{z} - D\vec{a}_i$.

We will explain later why (P) suffices for the correctness of the algorithm, how to choose an ϵ so that (P) holds, and why its description complexity is polynomial in n and b.

We start with justifying the algorithm's correctness, assuming that ϵ is chosen so

that (P) holds. We observe first that $\sum_i c_i \leq 1$ always. If the algorithm ever increases c, it is because $\vec{z} \neq \vec{a}_i$. If this is the case, then D from Step 12 will have some finite positive value. So $(1 - \frac{1}{1+D})(1-c) < 1 - c$, and adding c_i to c will not increase c past 1. We also observe that all the invariants declared in Step 2 hold throughout the course of the algorithm. This can be verified by simply checking each update rule in Step 14. Finally, we argue that every time the algorithm updates E, the dimension of $\bigcap_{h \in E} h$ decreases by 1, and $(\bigcap_{h \in E} h) \bigcap P \neq \emptyset$ is maintained. Because $\vec{a}_i$ and $\vec{z}$ both lie in $\bigcap_{h \in E}$ when Step 13 is executed, none of the hyperplanes in this intersection can possibly be violated at $(1+D+\epsilon)\vec{z} - (D+\epsilon)\vec{a}_i$. Therefore, the hyperplane output by $SO((1+D+\epsilon)\vec{z} - (D+\epsilon)\vec{a}_i)$ must reduce the dimension of $\bigcap_{h \in E} h$ by 1 when added to E at Step 14. Furthermore, because E_i is violated at $(1+D+\epsilon)\vec{z} - (D+\epsilon)\vec{a}_i$, but not at $(1+D)\vec{z} - D\vec{a}_i$, it must be the case that $(1+D)\vec{z} - D\vec{a}_i$ lies in the hyperplane E_i. (This holds because we will guarantee that our ϵ satisfies Property (P), described above.) Because this point is clearly in P, in the hyperplane E_i, and in all of the hyperplanes in E, it bears witness that we maintain $(\bigcap_{h \in E} h) \bigcap P \neq \emptyset$ always. Hence after at most n iterations of the while loop, the dimension of the remaining space is 0, and we must enter the case where $\vec{a}_i = \vec{z}$. The algorithm then exits outputting a convex combination of corners equaling $\vec{x}$.

It remains to argue that a choice of ϵ satisfying Property (P) is possible. Assuming the correctness of our algorithm, we show first that all the coefficients c_i computed by the algorithm have low bit complexity. Indeed, let $\vec{b_i} = (\vec{a}_i, 1)$ for all i. Once we know the algorithm is correct, the c_i's satisfy

$$\sum_i c_i \vec{b_i} = (\vec{x}, 1), \tag{2.1}$$

where c_i and $\vec{a}_i$ are outputs of our algorithm. We will argue that, for these $\vec{a}_i$s, the above system of linear equations has a unique solution. If not, let $\vec{c}$ and $\vec{c}\,'$ be two different solutions, and $d_i = c_i - c'_i$. We will show by induction on i that $d_i = 0$ for all i. In the base case, consider the hyperplane in E_1. We can write a corresponding $(n+1)$ dimensional vector $\vec{t_1}$, such that for all $\vec{x'} \in P$, $(\vec{x'}, 1) \cdot \vec{t_1} \leq 0$, and for all $i > 1$,

$\vec{b_i} \cdot \vec{t_1} = 0$. But $\vec{b_1} \cdot \vec{t_1} \neq 0$, otherwise, for any D, $(1+D)\vec{z} - D\vec{a}_1$ does not violate the constraint in E_1. On the other hand, $\sum_i d_i \vec{b_i} \cdot \vec{t_1} = 0$, therefore $d_1 = 0$. Now assume when $i < k$, $d_i = 0$, we will argue that $d_k = 0$. Let $\vec{t_k}$ be the corresponding vector for the hyperplane in E_k. For any $j > k$, $\vec{b_k} \cdot \vec{t_k} = 0$, and by the Inductive Hypothesis, for any $i < k$, $d_i = 0$, therefore $d_k \vec{b_k} \cdot \vec{t_k} = 0$. But we know $\vec{b_k} \cdot \vec{t_k} \neq 0$, otherwise, for any D, $(1+D)\vec{z} - D\vec{a_k}$ does not violate the constraint in E_k. So $d_k = 0$. Thus, $d_i = 0$ for all i.

So we have argued that the c_is are in fact the unique solution to the above linear system. We also know that the corners $\vec{a}_i$ (in fact all corners of the polytope) have poly(n, b) bit complexity. Applying the theory of Gaussian elimination, we deduce that each c_i can be described using no more than poly(n, b) bits, so the coefficients output by our algorithm have low bit complexity. Hence the $\vec{z}$ maintained by the algorithm has poly(n, b) bit complexity. So the intersections d_h of the ray $R(d) = \{(1+d)\vec{z} - d\vec{a}_i\}$ with the hyperplanes $h \in B(P)$ that do not contain both $\vec{z}$ and $\vec{a}_i$ (and hence the whole ray) also have poly(n, b) bit complexity. This guarantees that we can chose ϵ to be $2^{-\text{poly}(n,b)}$ to satisfy Property (P).

The above reasoning justifies the correctness of the algorithm. It is also now clear that every step runs in time polynomial in b, n, the runtime of SO and the runtime of CO, and each step is executed at most $n+1$ times. So the entire algorithm runs in polynomial time. □

Chapter 3

Feasibility of Single-Item Reduced Forms

In this chapter, we study the feasibility of single-item reduced form auctions. For a single item and independent (but not necessarily identically distributed) bidders, we show that any feasible reduced form auction can be implemented as a distribution over hierarchical mechanisms. We also give a polynomial-time algorithm for determining feasibility of a reduced form auction, or providing a separation hyperplane from the set of feasible reduced forms.

We overview our results in Section 3.1. We start with the special case when bidders are i.i.d., and the reduced form is bidder-symmetric. Then, we generalize our result to the case when bidders are independent but not identical. In the end, we provide a stronger characterization result for implementing a single-item feasible reduced form auction. We show that any single-item feasible reduced form auction can be implemented as a distribution over hierarchical mechanisms which respect the *same* ordering of types. Section 3.2 provides the full proof for the special case when bidders are i.i.d., and the reduced form is bidder-symmetric. Section 3.3 provides details for the case that bidders are independent. Section 3.4 provides the full proof for the strong characterization result for implementing single-item reduced forms.

3.1 Overview of Our Results

Single-item, Bidder-Symmetric Reduced Forms, i.i.d. Bidders

In the case of a single item and i.i.d. bidders, Border provided a necessary and sufficient condition for a bidder-symmetric reduced form to be feasible, generalizing prior partial results of Maskin-Riley [MR84] and Matthews [Mat84]. Let us review Border's theorem.

Theorem 3 ([Bor91])**.** *Suppose that the bidder's types are i.i.d. distributed according to some measure μ over T. Then a bidder-symmetric reduced form π is feasible if an only if*

$$\forall S \subseteq T: \quad m \cdot \int_S \pi(t) d\mu(t) \leq 1 - (1 - \mu(S))^m. \tag{3.1}$$

Simply put, a reduced form is feasible if and only if the probability that the item is awarded to a type in some set S (as computed by the reduced form) is at most the probability that someone with type from S shows up to the auction (as computed by the type distribution), for all subsets of types $S \subseteq T$. We call a set that violates this condition a *constricting set*. Clearly, the existence of a constricting set bears witness that the reduced form is infeasible, as the auctioneer cannot possibly award the item to someone in S if no one in S shows up. Border's theorem states that this is in fact a sufficient condition.

Border's original paper considered continuous type spaces (hence the integral in (3.1)), and the proof was based on measure theory. The following extension of the theorem was also shown: If there exists a constricting set S, then there is also a constricting set of the form S_x, where $S_x = \{A | \pi(A) > x\}$, for some x. In the case of finite type spaces, we can determine the feasibility of a reduced form auction in time $O(c \log c + c \cdot m)$, where $c = |T|$, as after sorting the type space in decreasing π's there are only c different subsets of the form S_x, and a dynamic program can find us if any of them violates (3.1) in time $O(c \cdot m)$. In other words, determining the feasibility of a bidder-symmetric reduced form, for a single item, and many i.i.d. bidders is

easy. However, the following important question was left unanswered: Given a feasible reduced form, can we efficiently obtain a mechanism implementing the reduced form? Notice that answering this question in the affirmative is absolutely necessary to be able to run the auction specified by the reduced form. Our first contribution is solving this problem.

Theorem 4. *Under the same assumptions as Theorem 3, given a bidder-symmetric single-item reduced form we can determine if it is feasible, or find a hyperplane separating it from the set of feasible bidder-symmetric reduced forms, in time* $O(c \cdot (\log c + m))$*, where* $c = |T|$*. If the reduced form is feasible, we provide a succinct description of a mechanism implementing the reduced form, in time polynomial in* $c \cdot m$*. The description of the mechanism is just (at most)* $c + 1$ *probabilities and an equal number of orderings of* T*.* [1] *The mechanism itself runs as follows: given the reported type profile, the mechanism samples a random subset of bidders in time polynomial in* $O(cm)$*, and the item is allocated uniformly at random to some bidder in that subset, or the item is thrown away.*

We prove Theorem 4 in Section 3.2, as a corollary of Proposition 2 and Theorem 2 of Section 3.2 and 2.8 respectively. In proving our result, we consider the following type of mechanisms:

Definition 4. *A* ***hierarchical mechanism*** *consists of a function* $H : T \to [c] \cup \{LOSE\}$*; one should interpret LOSE as a value larger than c. On bid vector* $(A_1, \ldots, A_m)$*, the mechanism has the following behavior: If* $H(A_i) = LOSE$ *for all* i*, the mechanism throws the item away. Otherwise, the item is awarded uniformly at random to a bidder in* $\operatorname{argmin}_i H(A_i)$*.*

In other words, a hierarchical mechanism breaks down the type space into a hierarchy. When the bidders arrive and submit their types, the mechanism finds the highest-priority level of the hierarchy that is populated by the submitted types, and gives the item uniformly at random to a bidder whose type falls in that level of the hierarchy (unless every bidder is a loser, in which case the mechanism throws the item away).

[1] An ordering may have equalities, but must be total (compare every pair of elements in T).

We say that a hierarchical mechanism H is *well-ordered* w.r.t. π if: $\pi(A) \geq \pi(A') \Rightarrow H(A) \leq H(A')$. We prove the following characterization result about feasible bidder-symmetric reduced forms:

Theorem 5. *When bidders are i.i.d., every feasible bidder-symmetric single-item reduced form π can be exactly implemented as a distribution over at most $c+1$ well-ordered with respect to π hierarchical mechanisms.*

Theorem 5 alone is not enough to allow us to implement a given bidder-symmetric reduced form. Indeed, if $\pi(\cdot)$ takes θ (can be as large as c) distinct values, there are 2^θ different well-ordered w.r.t. π hierarchical mechanisms. From here, we switch to our vector-view of reduced forms (as vectors $\vec{\pi}$ in $[0,1]^{|T|}$) and study the geometry of the space of feasible mechanisms respecting the order on the type-space induced by a given reduced form $\vec{\pi}$, which we will call P. We show that, in fact, P is a θ-dimensional polytope whose corners are *exactly* the 2^θ different well-ordered w.r.t. $\vec{\pi}$ hierarchical mechanisms. Using the geometric algorithm in Section 2.8 we can decompose $\vec{\pi}$ into a convex combination of at most $c+1$ corners of P in polynomial time. This convex combination is exactly a distribution over well-ordered w.r.t. $\vec{\pi}$ hierarchical mechanisms that implements $\vec{\pi}$. The geometric algorithm runs in time $\text{poly}(m, c)$, and sampling from the distribution output by our algorithm takes time $O(c)$. We provide the details of our approach and proofs of the relevant claims in Section 3.2 and 2.8.

Single-item Reduced Forms, Non-i.i.d. Bidders

Recently, an alternative proof of Border's theorem for distributions with finite support was discovered in [Bor07] and again in [CKM11], the latter using a clean network-flow interpretation. These proofs extend Theorem 3 to independent, but not necessarily identical, bidders and non-symmetric reduced forms. In this case, (3.1) is replaced by the following necessary and sufficient condition:

$$\forall S_1 \subseteq T_1, \ldots, S_m \subseteq T_m : \sum_i \sum_{A \in S_i} \pi_i(A) \Pr[t_i = A] \leq 1 - \prod_i (1 - \Pr[t_i \in S_i]). \quad (3.2)$$

The interpretation of the LHS and RHS of the above inequality is the same as the one given above for (3.1) except generalized to the non-iid non-symmetric setting. In addition to the above condition, [CKM11] proves a generalization of Border's extended result: If there is a constricting $S = (S_1, \ldots, S_m)$, then there is also a constricting set of the form $S' = (S_{x_1}^{(1)}, \ldots, S_{x_m}^{(m)})$, where $S_{x_i}^{(i)} = \{A \in T_i | \pi_i(A) > x_i\}$. In other words, each bidder has a different threshold x_i, and $S_{x_i}^{(i)}$ contains all types of bidder i with π_i above x_i. Unfortunately, despite this simplification, there are still $\prod_i (|T_i|+1)$ possible constricting sets, and testing each of them would take time exponential in the number of bidders.

One might hope to obtain a stronger theorem that would only require testing a number of sets polynomial in c and m. We prove such a theorem by introducing a notion of a *virtual* π, defined next. We name it such not because the equation involves hazard rates or looks anything like that for virtual valuations [Mye81], but because the spirit of the transformation is the same. Myerson observed that he could make the most revenue not from the bidder with the highest valuation, but from the bidder with the highest virtual valuation. Likewise, in our setting, the most difficult types to satisfy are not the types with the highest π, but the types with the highest virtual π. The definition of virtual π, which we denote $\hat{\pi}$, is actually quite simple.

Definition 5. *If π is a reduced form, we define its corresponding* virtual reduced form $\hat{\pi}$ *as follows: for all i and type $A \in T_i$, $\hat{\pi}_i(A) := \Pr[\pi_i(t_i) \leq \pi_i(A)]\pi_i(A)$.*

It turns out that this definition exactly captures which types of different bidders are harder to satisfy. In the bidder-symmetric case of Section 3.1, we were able to compare a pair of types A and B submitted by bidders $i \neq k$ based only on their corresponding $\pi_i(A)$ and $\pi_k(B)$. This is no longer the case in the non-iid case, resulting in the more complicated constricting sets defined above. Nevertheless, we show that A and B can be compared at face value of $\hat{\pi}_i(A)$ and $\hat{\pi}_k(B)$:

Theorem 6. *Suppose that the bidders are independent and there is a single item for sale. A reduced form π is feasible if and only if: for all x, the sets $S_x^{(i)} = \{A \in$*

$T_i|\hat{\pi}_i(A) > x\}$ *satisfy:*

$$\sum_i \sum_{A \in S_x^{(i)}} \pi_i(A) \Pr[t_i = A] \leq 1 - \prod_i (1 - \Pr[t_i \in S_x^{(i)}]). \tag{3.3}$$

In particular, we can test the feasibility of a reduced form, or obtain a hyperplane separating the reduced form from the set of feasible reduced forms, in time linear in $\sum_i |T_i| \cdot (\log(\sum_i |T_i|) + m)$.

Details of the proof can be found in Section 3.3. We also prove there two analogues of Theorem 5 in this setting: Theorem 7, which is used for our algorithmic results, and Theorem 9, which provides a much stronger characterization. The analog of Definition 4 is the following:

Definition 6. *A* ***hierarchical mechanism*** *consists of a function* $H : \bigcup_i (T_i \times \{i\}) \to [\sum_i |T_i|] \cup \{LOSE\}$*; one should interpret LOSE as a value larger than* $\sum_i |T_i|$*. On bid vector* $(A_1, \ldots, A_m)$*, if* $H(A_i, i) = LOSE$ *for all* i*, the mechanism throws the item away. Otherwise, the item is awarded uniformly at random to a bidder in* $\operatorname{argmin}_i H(A_i, i)$.

We say that a hierarchical mechanism H for non-identical bidders is *partially-ordered* w.r.t. π if for all i and $A, A' \in T_i$, $\pi_i(A) \geq \pi_i(A') \Rightarrow H(A, i) \leq H(A', i)$. We say that a hierarchical mechanism is *strict* if for all bidders i, j and types $A \in T_i$, $B \in T_j$: $i \neq j \Rightarrow (H(A, i) \neq H(B, j) \vee H(A, i) = H(B, j) = \text{LOSE})$ (i.e. there is always a unique winner in $\operatorname{argmin}_i H(A_i, i)$ if one exists, because each level (except possibly for LOSE) contains types from only a single bidder). Our algorithmic extension of Theorem 5 is the following:

Theorem 7. *When bidders are independent, but not necessarily identically distributed, every feasible single-item reduced form* π *can be exactly implemented as a distribution over at most* $\sum_i |T_i| + 1$ *strict, partially-ordered w.r.t.* π *hierarchical mechanisms.*

From here, we take the same geometric approach as in Section 3.1 and study the geometry of the set of feasible reduced forms that respect the partial-ordering of types

induced by a given reduced-form π. Again we show that this is a $(\sum_i d_i)$-dimensional polytope, P, where d_i (could be as large as $|T_i|$) is the number of distinct values that $\pi_i(\cdot)$ takes on input from T_i, and that the corners of P are *exactly* the strict, partially-ordered w.r.t. π hierarchical mechanisms. Writing a point in P as a convex combination of $\sum_i |T_i| + 1$ corners is no longer an easy procedure. Not only does P have an exponential number of corners, but there are also exponentially many hyperplanes defining the boundary of P (where there were only $2 \cdot c$ such hyperplanes in the i.i.d. case). Luckily, Theorem 6 provides an efficient separation oracle for membership in P. By making use of this separation oracle instead of checking the exponentially-many boundary equations one by one, the geometric algorithm of Section 2.8 outputs a representation of a given π as a convex combination of at most $\sum_i |T_i| + 1$ corners of P, which is exactly a distribution over the corresponding $\sum_i |T_i| + 1$ strict, partially-ordered w.r.t. π hierarchical mechanisms. Putting this approach together with Theorems 6 and 7, we obtain in Section 3.3 the algorithmic result of this section:

Theorem 8. *When bidders are independent, given a single-item reduced form we can determine if it is feasible, or find a hyperplane separating it from the set of feasible reduced forms, in time linear in* $\sum_i |T_i| \cdot (\log(\sum_i |T_i|) + m)$. *If the reduced form is feasible, we can compute a succinct description of a mechanism implementing the reduced form, in time polynomial in* $\sum_i |T_i|$. *The description of the mechanism is just (at most)* $\sum_i |T_i| + 1$ *probabilities and the same number of total orderings of* $\bigcup_i (T_i \times \{i\})$. *The mechanism itself runs as follows: given the reported type profile, the mechanism samples a random total ordering of all bidders' types in time* $O(\sum_i |T_i|)$, *and allocates the item to the bidder whose reported type is highest in that ordering, or throws the item away.*

While Theorem 7 does provide some structure to the otherwise unknown set of feasible mechanisms for independent bidders, the result is not as compelling as that of Theorem 5. One might have hoped that every feasible reduced form can be implemented as a distribution over *virtually-ordered* hierarchical mechanisms (that is, hierarchical mechanisms such that $\hat{\pi}_i(A) \geq \hat{\pi}_j(B) \Rightarrow H(A, i) \leq H(B, j)$). Unfortu-

nately, this is not true, as is shown in Section 3.3. Despite this, we show that a strong generalization of Theorem 5 holds in this setting. Let σ be a total ordering on the elements of $\bigcup_i (T_i \times \{i\})$ (i.e. a mapping $\sigma : \bigcup_i (T_i \times \{i\}) \to [\sum_i |T_i|]$). We say that σ respects π if $\pi_i(A) > \pi_i(B) \Rightarrow \sigma(A,i) < \sigma(B,i)$. We also say that a hierachical mechanism H is σ-ordered if $\sigma(A,i) < \sigma(B,j) \Rightarrow H(A,i) \leq H(B,j)$. We prove the following theorem:

Theorem 9. *If a single-item reduced form π is feasible, there exists a total ordering σ on the elements of $\bigcup_i (T_i \times \{i\})$ that respects π such that π can be implemented as a distribution over σ-ordered hierarchical mechanisms.*

We provide the proof of Theorem 9 in Section 3.4.

3.2 Single-item, I.I.D. Bidders, Bidder-Symmetric Reduced Forms

We provide the details and proofs of the techniques discussed in Section 3.1 for single-item, i.i.d. bidders, and bidder-symmetric reduced forms, beginning with some technical lemmas.

Lemma 1. *Every feasible bidder-symmetric reduced form auction π for i.i.d. bidders and a single item can be implemented (not necessarily exactly implemented) as a distribution over well-ordered w.r.t. π hierarchical mechanisms.*

Proof. Our proof is by induction on the number of distinct non-zero values in $\{\pi(A)\}_{A \in T}$.

Base Case: There is a single non-zero value in this set, equal to some $x \leq 1$. The mechanism that gives the item uniformly at random to a bidder whose reported type A satisfies $\pi(A) = x$ (if such bidder shows up) implements this reduced form as long as it is feasible.

Inductive Hypothesis: For all $0 < k < \theta$, every feasible reduced form with k distinct non-zero values in $\{\pi(A)\}_{A \in T}$ can be implemented as a distribution over well-ordered w.r.t. π hierarchical mechanisms.

Inductive Step: We show that the inductive hypothesis extends to the case where there are exactly θ distinct non-zero values in $\{\pi(A)\}_{A \in T}$. Let X denote the set of all distributions over well-ordered w.r.t. π hierarchical mechanisms. Then X can be interpreted as a closed, bounded subset of $\mathbb{R}^{2^\theta}$, where each coordinate denotes the probability of using one of the 2^θ well-ordered w.r.t. π hierarchical mechanisms. Therefore, X is compact. For a distribution over hierarchical mechanisms $M \in X$, denote by $M(A)$ the probability that a bidder reporting type A receives the item under M. Define the function $F : X \to \mathbb{R}$ as:

$$F(M) = \max_A \{\pi(A) - M(A)\}.$$

Let us use the Euclidean distance in X as a subset of $\mathbb{R}^{2^\theta}$. As X is a compact space, and F is a continuous function, F attains its minimum in X. Let M denote one such minimizer of F. Then if $F(M) \leq 0$, M implements the reduced form. If $F(M) > 0$, we will show a contradiction. Let S denote the subset of types $\text{argmax}_A\{\pi(A) - M(A)\}$, i.e. the subset of types who are the most unsatisfied by M.

We show first that, if S contains every non-zero type, then the reduced form is infeasible. We may, w.l.o.g., assume that M always awards the item to a non-zero type if one shows up, as this will not decrease $M(A)$ for any non-zero A. Therefore, we know that

$$\sum_{A:\pi(A)\neq 0} \Pr[A] M(A) = \Pr[\text{see a non-zero type}].$$

However, if $\pi(A) - M(A) > 0$ for all non-zero types, then we must have

$$\sum_{A:\pi(A)\neq 0} \Pr[A]\pi(A) > \Pr[\text{see a non-zero type}]$$

and the reduced form is infeasible. So if the reduced form is feasible, S must be missing at least one non-zero type.

Now let $s = |\{\pi(A)\}_{A \in S}|$ be the number of distinct non-zero values assigned by π to types in S. We argue that $s < \theta$. To see this, it suffices to observe the following: for all types B and B', $\pi(B) = \pi(B')$ implies that $M(B) = M(B')$ (this is because, by

definition, all hierarchical mechanisms H in the support of M satisfy $H(B) = H(B'))$. So in particular either $B, B' \in S$ or $B, B' \notin S$, but it cannot be that one of B, B' is in S and the other is not. And because S is missing at least one non-zero type, $s < \theta$.

To show a contradiction to $F(M) > 0$, let us define a new reduced form π' as follows. For all $A \in S$, set $\pi'(A) = \pi(A)$. For all $A \notin S$, set

$$\pi'(A) = \max_{B \in S | \pi(B) < \pi(A)} \{\pi(B)\},$$

unless $\{B \mid B \in S \wedge \pi(B) < \pi(A)\}$ is empty in which case we set $\pi'(A) = 0$. Observe that the number of distinct non-zero values in $\{\pi'(A)\}_{A \in T}$ is exactly $s < \theta$. So it follows by our inductive hypothesis that π' can be implemented by a distribution over well-ordered (with respect to π') hierarchical mechanisms. In fact, as $\pi(A) \geq \pi(A') \Rightarrow \pi'(A) \geq \pi'(A')$, every hierarchical mechanism that is well-ordered with respect to π' is also well-ordered with respect to π. Call M' the distribution over well-ordered hierarchical mechanisms implementing π'. Now, set $\epsilon = (F(M) - \text{argmax}_{A \notin S}\{\pi(A) - M(A)\})/2$, and consider the distribution $M'' = (1 - \epsilon)M + \epsilon M'$ (with probability $(1 - \epsilon)$ sample from M, with probability ϵ sample from M').

What is $M''(A)$? If $A \in S$, then $M'(A) = \pi(A)$, so $M''(A) = (1-\epsilon)M(A)+\epsilon\pi(A)$. So for all $A \in S$, $M''(A) > M(A)$, hence $\pi(A) - M''(A) < F(M)$.

If $A \notin S$, then $M'(A) \geq 0$, so $M''(A) \geq (1 - \epsilon)M(A) \geq M(A) - \epsilon$, so we get $\pi(A) - M''(A) \leq \pi(A) - M(A) + \epsilon < F(M)$. Putting both observations together, we see that $F(M'') < F(M)$, a contradiction.

So we must have $F(M) \leq 0$, meaning M implements the reduced form, completing the inductive step and the proof of the lemma. □

Corollary 1. *Every feasible bidder-symmetric reduced form π can be exactly implemented as a distribution over well-ordered w.r.t. π hierarchical mechanisms.*

Proof. It follows from Lemma 1 that π can be implemented as a distribution over well-ordered w.r.t. π hierarchical mechanisms. Let then X denote the set of distributions over well-ordered w.r.t. π hierarchical mechanisms that implement π. As in Lemma 1 the set X, viewed as a subset of $\mathbb{R}^{2^\theta}$, where θ is the number of distinct non-zero values

in π, is compact. We can also define the function $G : X \to \mathbb{R}$ as:

$$G(M) = \max_A \{M(A) - \pi(A)\}.$$

Equipping X with the Euclidean distance of $\mathbb{R}^{2^\theta}$, G is a continuous function on X. As X is compact and G continuous, G attains its minimum in X.

We show that the minimum of G is exactly 0 (i.e. that a minimizer of G exactly implements π), following an induction similar to the one used in Lemma 1 in terms of the number of distinct non-zero values in $\{\pi(A)\}_{A \in T}$. We sketch the steps involved for the inductive step. Take any minimizer M of G. If $G(M) \leq 0$, then because M has to implement π, M must exactly implement π. If $G(M) > 0$, then let $S = T - \text{argmax}_A\{M(A) - \pi(A)\}$. Then, for all $A \in S$, define $\pi'(A) = \pi(A)$. For all $A \notin S$, define $\pi'(A) = \max_{B \in S | \pi(B) \leq \pi(A)}\{\pi(B)\}$, unless $\{B : B \in S \wedge \pi(B) \leq \pi(A)\}$ is empty, in which case set $\pi'(A) = 0$. As $\text{argmax}\{M(A) - \pi(A)\}$ can't possibly be empty the number of distinct non-zero values in $\{\pi'(A)\}_{A \in T}$ is smaller than that in $\{\pi(A)\}_{A \in T}$. (We still use the observation that, for all types B and B', $\pi(B) = \pi(B')$ implies that $M(B) = M(B')$). The rest of the inductive step proceeds identically with that in the proof of Lemma 1, resulting in a contradiction. Hence, it can't be that the minimizer of G satisfies $G(M) > 0$. □

To proceed we need a definition. Let $> := A_1 > A_2 > \ldots > A_k$ be a total ordering of a set $T_k := \{A_1, \ldots, A_k\}$ of k types, and consider a setting where we have m bidders whose types are distributed i.i.d. over T_k. We say that a bidder-symmetric reduced form $\pi : T_k \to [0, 1]$ respects $>$ iff $\pi(A_1) \geq \pi(A_2) \geq \ldots \geq \pi(A_k)$. We also say that an hierarchical mechanism H on T_k is well-ordered with respect to $>$ iff $H(A_1) \leq H(A_2) \leq \ldots \leq H(A_k)$. [2] With respect to these definitions we show the following proposition.

[2]Notice that this definition of well-ordered hierarchical mechanism (with respect to $>$) is very similar to its counterpart in the main body (with respect to π), but different. Being well-ordered with respect to π certainly imposes the same constraints as being well ordered with respect to any $>$ that π respects. The difference is that being well-ordered with respect to π may also impose some equality constraints, if $\pi(A) = \pi(B)$ for types $A \neq B$.

Proposition 2. *Consider m bidders whose types are distributed i.i.d. over $T_k := \{A_1, \ldots, A_k\}$ and a total ordering $>:= A_1 > A_2 > \ldots > A_k$ of T_k. The set of feasible bidder-symmetric reduced forms that respect $>$ is a k-dimensional polytope whose corners are exactly the 2^k hierarchical mechanisms that are well-ordered with respect to $>$.*

Proof. As a corollary of Theorem 3, a reduced form respects $>$ and is feasible if and only if

$$\pi(A_i) \geq \pi(A_{i+1}) \qquad \forall i \in [k]; \tag{3.4}$$

$$\sum_{j \leq i} m \cdot \Pr[A_j]\pi(A_j) \leq 1 - \left(1 - \sum_{j \leq i} \Pr[A_j]\right)^m \qquad \forall i \in [k]; \tag{3.5}$$

where for notational convenience we denote $\pi(A_{k+1}) = 0$. We have hence shown that the set of feasible bidder-symmetric reduced forms that respect $>$ is a k-dimensional polytope.

We proceed to show that each well-ordered w.r.t. $>$ hierarchical mechanism is a corner. Let H be such a mechanism and π be the reduced form that it induces. Then, for all i (including $i = k$, denoting $H(A_{k+1}) = \text{LOSE}$) we either have $H(A_i) = H(A_{i+1})$, in which case $\pi(A_i) = \pi(A_{i+1})$, or $H(A_i) < H(A_{i+1})$, in which case $\sum_{j \leq i} m \cdot \Pr[A_j]\pi(A_j) = 1 - (1 - \sum_{j \leq i} \Pr[A_j])^m$, because the item is always awarded to one of the top i types whenever one is present. Therefore, at least k of the inequalities defining the polytope are tight. And it is easy to see that there is a unique reduced form making these inequalities tight. It is also clear that every well-ordered w.r.t. $>$ hierarchical mechanism is inside the polytope. So every well-ordered w.r.t. $>$ hierarchical mechanism is a corner of the polytope.

Finally, we show that there are no other corners. Assume for contradiction that there was a corner π of the polytope that is not a well-ordered w.r.t. $>$ hierarchical mechanism. Then by Corollary 1, we know that π can be written as a convex combination of well-ordered w.r.t. π hierarchical mechanisms, and hence as a convex combination of well-ordered w.r.t. $>$ hierarchical mechanisms. (As π respects $>$ a

hierarchical mechanism that is well-ordered w.r.t. π will also be well-ordered w.r.t. $>$). As every well-ordered w.r.t. $>$ hierarchical mechanism is a corner of the polytope, and π is not one of them, this means that π can be written as a convex combination of *other* corners of the polytope, which contradicts that π is itself a corner.

Therefore we have shown that every feasible bidder-symmetric reduced form respecting $>$ lies inside the afore-described polytope, every well-ordered w.r.t. $>$ hierarchical mechanism is a corner of this polytope, and there are no other corners. This establishes the proposition. □

Now we can put everything together to prove Theorems 5 and 4.

Proof of Theorem 5: Suppose that the bidders' types are sampled i.i.d. from T according to $\mathcal{D}_1$, and let π be a feasible bidder-symmetric reduced form. We do the following preprocessing operation on the set of types T:

> TypeMerge: Find a maximal set of types $A_1, \ldots, A_\ell \in T$ such that $\pi(A_1) = \ldots = \pi(A_\ell)$. Then remove types $A_1, \ldots, A_\ell$ from T and add super-type $\langle A_1, \ldots, A_\ell \rangle$ into T; change $\mathcal{D}_1$ to never sample any of $A_1, \ldots, A_\ell$ and sample the super-type $\langle A_1, \ldots, A_\ell \rangle$ with probability $\sum_i \Pr[A_i]$; and set $\pi(\langle A_1, \ldots, A_\ell \rangle) = \pi(A_1)$. Repeat this procedure until every type in T has different π value, i.e. until the set $\{\pi(A)\}_{A \in T}$ has cardinality $|T|$.

Let T', π', $\mathcal{D}'_1$ be the type-set, reduced-form, type-distribution resulting from the TypeMerge operation on input π. We claim that:

1. π' is a feasible bidder-symmetric reduced form for bidders sampled i.i.d. from $\mathcal{D}'_1$. This follows immediately from the feasibility of π. Indeed it follows from Theorem 3 and our discussion in Section 3.1 that a sufficient condition for the feasibility of π' is for it to satisfy Eq. (3.1) for all subsets of types of the form $\{A \mid A \in T' \wedge \pi'(A) \geq x\}$. On the other hand the feasibility of π implies that π satisfies Eq. (3.1) for all subsets of types of the form $\{A \mid A \in T \wedge \pi(A) \geq x\}$. This together with the nature of our TypeMerge operation implies that π' satisfies the afore-mentioned sufficient conditions for feasibility.

2. A mechanism that exactly implements π' immediately gives a mechanism exactly implementing π. Indeed, to implement π we just run the mechanism implementing π' after replacing in the reported type vector every type A that was removed from T by TypeMerge by its corresponding super-type.

3. A hierarchical mechanism H' that is well-ordered w.r.t. π' can be *expanded* into a hierarchical mechanism H that is well-ordered w.r.t. π. Indeed, if super-type $\langle A_1, \ldots, A_\ell \rangle$ replaced types $A_1, \ldots, A_\ell$ during the TypeMerge operation we set $H(A_i) := H'(\langle A_1, \ldots, A_\ell \rangle)$, for all $i = 1, \ldots, \ell$. On the other hand, if a type A belongs in both T and T', we set $H(A) := H'(A)$. Moreover, if $\pi_{H'}$, π_H are respectively the reduced forms induced by H' and H, the following property is satisfied. If super-type $\langle A_1, \ldots, A_\ell \rangle$ replaced types $A_1, \ldots, A_\ell$ during the TypeMerge operation, then $\pi_H(A_i) := \pi_{H'}(\langle A_1, \ldots, A_\ell \rangle)$, for all $i = 1, \ldots, \ell$. On the other hand, if a type A belongs in both T and T', then $\pi_H(A) := \pi_{H'}(A)$.

Now, suppose that the cardinality of T' is k. Given that π' assigns a distinct probability to every type in T', it induces a total ordering on T'. In particular, suppose that $T' := \{A_1, \ldots, A_k\}$, where $\pi'(A_1) > \pi'(A_2) > \ldots > \pi'(A_k)$. By Proposition 2, π' lies inside a k-dimensional polytope whose corners are exactly the 2^k hierarchical mechanisms that are well-ordered with respect to the order $>:= A_1 > \ldots > A_k$. By Carathéodory's Theorem, every point in the polytope can be written as a convex combination of at most $k+1$ corners. As a convex combination of corners is exactly a distribution over well-ordered hierarchical mechanisms w.r.t. $>$, we get that π' can be written as a distribution over hierarchical mechanisms that are well-ordered w.r.t. $>$, and hence also w.r.t. π'. Now we can expand all hierarchical mechanisms in the support of the distribution, according to the procedure described in Step 3 above, to obtain that π can be written as a distribution over hierarchical mechanisms that are well-ordered w.r.t. π. $\square$

Proof of Theorem 4: It follows from our discussion in Section 3.1 that a bidder-symmetric reduced form $\tilde{\pi}$ is infeasible if and only if it violates Eq. (3.1) for a subset

of types of the form $\{A \mid A \in T \wedge \tilde{\pi}(A) \geq x\}$. Since there are at most $c \equiv |T|$ such sets we can efficiently determine feasibility of a given reduced from $\tilde{\pi}$ or provide a hyperplane separating it from the set of feasible reduced forms.

We now need to describe how to efficiently find a mechanism implementing a reduced form $\tilde{\pi}$ that is feasible. In view of the TypeMerge operation defined in the proof of Theorem 5, we can w.l.o.g. assume that $\tilde{\pi}$ assigns a distinct probability to every type in T. (Otherwise we can always run TypeMerge to merge types sharing the same $\tilde{\pi}$-probability to super-types and apply the procedure outlined below to the output of the TypeMerge operation, and then go back to the original $\tilde{\pi}$). Under the assumption that $\tilde{\pi}$ assigns distinct probabilities to all types in T, Proposition 2 implies that $\tilde{\pi}$ lies inside a c-dimensional polytope, P, whose corners are the well-ordered w.r.t. $\tilde{\pi}$ hierarchical mechanisms. Therefore we can directly apply Theorem 2 of Section 2.8 to write $\tilde{\pi}$ as a convex combination of such hierarchical mechanisms, as long as we can describe the boundary oracle BO, corner oracle CO and separation oracle SO that are needed for Theorem 2. BO is trivial to implement, as we just have to include in the set of halfspaces defining the boundary of P those inequalities described in the proof of Proposition 2. For CO, on input B, we first check that every hyperplane $h \in B$ satisfies $BO(h) =$ yes. If not, output no. Otherwise we need to check if $\bigcap_{h \in B} h$ contains a corner of P. We know that the corners of P are exactly the well-ordered w.r.t. $\tilde{\pi}$ hierarchical mechanisms. So none of the corners lies in the intersection of the hyperplanes $\pi(A_i) = \pi(A_{i+1})$ and $\sum_{j \leq i} m \cdot \Pr[A_j]\pi(A_j) = 1 - (1 - \sum_{j \leq i} \Pr[A_j])^m$, for any i. (Indeed, for a hierarchical mechanism H and its induced reduced form π, $\pi(A_i) = \pi(A_{i+1})$ implies that $H(A_i) = H(A_{i+1})$, yet $\sum_{j \leq i} m \cdot \Pr[A_j]\pi(A_j) = 1 - (1 - \sum_{j \leq i} \Pr[A_j])^m$ implies $H(A_i) > H(A_{i+1})$). So, if B contains any pair of hyperplanes of this form, output no. Otherwise, for all i such that $\pi(A_i) = \pi(A_{i+1}) \in B$, set $H(A_i) = H(A_{i+1})$, otherwise set $H(A_i) = H(A_{i+1}) - 1$. This defines a well-ordered w.r.t. $\tilde{\pi}$ hierarchical mechanism that is in $\bigcap_{h \in B} h$, so have CO output H. Finally, SO is easy to implement as we can just check each of the $2 \cdot c$ inequalities written in the proof of Proposition 2 one by one.

So because we can implement BO, CO, SO in polynomial time, we can apply

Theorem 2 to write $\tilde{\pi}$ as a convex combination of at most $c+1$ corners, which is exactly a distribution over at most $c+1$ well-ordered w.r.t. $\tilde{\pi}$ hierarchical mechanisms in polynomial time. $\square$

3.3 Single-item, Independent Bidders

Here we provide details of Section 3.1 for the case of single-item independent bidders, and the proofs of Theorems 6, 7 and 8, postponing the proof of Theorem 9 to Section 3.4. Before proving our theorems, we show that the concept of virtual πs is necessary. As in, Theorem 6 would be false if we tried to replace $\hat{\pi}$ with π.

Proposition 3. *There exist reduced forms that are infeasible, yet for all S_x^i of the form $S_x^i = \{A \mid \pi_i(A) > x,\ \forall i\}$:*

$$\sum_i \sum_{A \in S_x^i} \pi_i(A) \Pr[t_i = A] \leq 1 - \prod_{i=1}^{m} (1 - \sum_{A \in S_x^i} \Pr[t_i = A]).$$

Proof. Consider the case with two bidders. Bidder 1 has two types, with $\Pr[t_1 = A] = 1/8$, $\Pr[t_1 = B] = 7/8$, $\pi_1(A) = 5/8$, $\pi_1(B) = 0$. Bidder 2 has two types, with $\Pr[t_2 = C] = 1/2$, $\Pr[t_2 = D] = 1/2$, $\pi_2(C) = 1$, $\pi_2(D) = 3/4$.

Then this reduced form is infeasible. Indeed, observe that C must always receive the item whenever $t_2 = C$, which happens with probability $1/2$. So if se have $\pi_2(C) = 1$, we cannot also have $\pi_1(A) > 1/2$. So the set $\{A, C\}$ forms a constricting set. However, the sets of the form S_x^i are $\{C\}, \{C, D\}, \{C, D, A\}, \{C, D, A, B\}$, and they all satisfy the above inequality. □

Proposition 3 shows us that ordering the types of all bidders by decreasing π doesn't allow us to correctly determine the feasibility of a reduced form. Similarly, a partial ordering of the types that only orders a single bidder's types by decreasing π doesn't give enough structure to efficiently determine the feasibility of the reduced form. What we need is a correct total ordering of the types of all bidders, and we can obtain it using virtual πs. Here is a quick observation about the virtual πs, followed by a proof of Theorem 6.

Observation 1. *For two types $A, B \in T_i$, $\hat{\pi}_i(A) \geq \hat{\pi}_i(B) \Leftrightarrow \pi_i(A) \geq \pi_i(B)$.*

Proof. If $\pi_i(A) \geq \pi_i(B)$, then $\Pr[\pi_i(t_i) \leq \pi_i(A)] \geq \Pr[\pi_i(t_i) \leq \pi_i(B)]$. Therefore, $\hat{\pi}_i(A) \geq \hat{\pi}_i(B)$. The other direction is identical. □

Proof of Theorem 6: We know from [Bor07, CKM11], that if a reduced form mechanism is infeasible, then there is some constricting set of the form $S = \bigcup_{i=1}^{m} S_{x_i}$, where $S_{x_i} = \{A \mid \pi_i(A) \geq x_i, A \in T_i\}$. (Forgive the abuse of notation here. Formally, S is a collection of m sets of types, one for each bidder. To avoid cumbersome notation and take union casually in this proof, let us assume that a type $A \in T_i$ carries the name of bidder i, for all i.) Now consider any minimal constricting set of this form, i.e. a choice of $x_1, \ldots, x_m$ such that replacing S_{x_i} with $S_{x_i} - \{A\}$ ($A \in S_{x_i}$) results in S no longer being a constricting set. [3] Now let $(i, A) \in \operatorname{argmin}_{i,\ A \in S_{x_i}} \hat{\pi}_i(A)$. Then by Observation 1 and by our choice of S, $S - \{A\}$ is not a constricting set. Therefore, adding A to $S - \{A\}$ must increase the left-hand bound by more than it increases the right-hand bound:

$$\Pr[t_i = A]\pi_i(A) > \Pr[t_i = A] \prod_{j \neq i} \Pr[\pi_j(t_j) < x_j]$$

$$\implies \frac{\pi_i(A)}{\prod_{j \neq i} \Pr[\pi_j(t_j) < x_j]} > 1.$$

Now consider any other $A' \in T_k$, $A' \notin S$ and $\hat{\pi}_k(A') \geq \hat{\pi}_i(A)$. Observe first that we must have A' from some bidder $k \neq i$, as every $A'' \in T_i$ with $\hat{\pi}_i(A'') \geq \hat{\pi}_i(A)$ has $\pi_i(A'') \geq \pi_i(A) \geq x_i$, so we would have $A'' \in S$. So for this A', we have:

$$\begin{aligned}
&\pi_k(A') \Pr[\pi_k(t_k) \leq \pi_k(A')] \geq \pi_i(A) \Pr[\pi_i(t_i) \leq \pi_i(A)] \\
\implies &\pi_k(A') \Pr[\pi_k(t_k) < x_k] \geq \pi_i(A) \Pr[\pi_i(t_i) < \pi_i(A)] \\
\implies &\pi_k(A') \Pr[\pi_k(t_k) < x_k] \geq \pi_i(A) \Pr[\pi_i(t_i) < x_i] \\
\implies &\pi_k(A') \prod_{j \neq i} \Pr[\pi_j(t_j) < x_j] \geq \pi_i(A) \prod_{j \neq k} \Pr[\pi_j(t_j) < x_j] \\
\implies &\frac{\pi_k(A')}{\prod_{j \neq k} \Pr[\pi_j(t_j) < x_j]} \geq \frac{\pi_i(A)}{\prod_{j \neq i} \Pr[\pi_j(t_j) < x_j]}.
\end{aligned}$$

[3] For a minimal set S, there could be many possible choices of $x_1, \ldots, x_m$. We simply use any of them.

And by our choice of A and the work above, we obtain:

$$\frac{\pi_k(A')}{\prod_{j\neq k}\Pr[\pi_j(t_j) < x_j]} > 1$$

$$\implies \Pr[t_k = A']\pi_k(A') > \Pr[t_k = A']\prod_{j\neq k}\Pr[\pi_j(t_j) < x_j].$$

This equation tells us directly that we could add A' to S and still get a constricting set. In fact, it tells us something stronger. If $S' = \bigcup_j S'_j$, where $S'_j \subseteq T_j$, is any constricting set containing S, then we could add A' to S' and still have a constricting set. This is because the change to the left-hand side of the inequality is the same, no matter what set we are adding A' to. It is always $\Pr[t_k = A']\pi_k(A')$. And the change to the right-hand side is exactly $\Pr[t_k = A']$ times the probability that none of the types in $\cup_{j\neq k}S'_j$ show up. As we add more types to S, the probability that none of the types in $\cup_{j\neq k}S'_j$ show up will never increase. So for any constricting set S' containing S, we can add A' to S'_k and still get a constricting set.

So starting from a constricting set S and a type $A \in T_i$ as above we can add every $B \in T_j$ with $\hat{\pi}_j(B) \geq \hat{\pi}_i(A)$ to S in order to obtain a constricting set of the form $S_x = \{B | B \in T_j \wedge \hat{\pi}_j(B) \geq x\}$, where $x = \hat{\pi}_i(A)$. So every infeasible reduced form has a constricting set of this form. Taking the contrapositive proves the theorem. □

We say that a hierarchical mechanism H is *virtually-ordered* if $\hat{\pi}_i(A) \geq \hat{\pi}_j(A') \Rightarrow H(A, i) \leq H(A', j)$. While the virtual πs give us a nice structural theorem about feasible reduced forms and a linear time separation oracle for determining feasibility (see proof of Theorem 8), the following observation shows that distributions over virtually-ordered hierarchical mechanisms are not sufficient to implement every feasible reduced form when the bidders are non-i.i.d.

Observation 2. *There exist feasible reduced forms that are not implementable as distributions over virtually-ordered hierarchical mechanisms.*

Proof. Consider the following example with two bidders. Bidder one has a single type, A. Bidder two has two types, B and C and is each with probability $1/2$. Then

$\pi_1(A) = 1/3$, $\pi_2(B) = 2/3 + \epsilon$, $\pi_2(C) = 2/3 - \epsilon$ is a feasible reduced form. However, $\hat{\pi}_1(A) > \hat{\pi}_2(C)$, so no distribution over virtually-ordered hierarchical mechanisms can possibly have $\pi_2(C) > 1/2$. □

Now that we have motivated Theorems 7 and 8, we proceed to prove them, after providing the key steps as technical lemmas.

Lemma 2. *Every feasible reduced form π for independent bidders and a single item can be implemented (not necessarily exactly implemented) as a distribution over strict, partially-ordered w.r.t. π hierarchical mechanisms.*

Proof. The proof is almost identical to that of Lemma 1. Here are the main differences: We do induction on $\sum_i d_i$, where d_i is the number of distinct non-zero values in $\{\pi_i(A)\}_{A \in T_i}$. For the inductive step, X is now taken to be the set of distributions over strict, partially-ordered hierarchical mechanisms, and it is still compact, viewed as a subset of the Euclidean space. The function $F : X \to \mathbb{R}$ is now defined as

$$F(M) = \max_{i, A \in T_i} \{\pi_i(A) - M_i(A)\}.$$

Again, if we use the Euclidean distance on X, as a subset of the Euclidean space, F is continuous. Since F is continuous and X is compact, F achieves its minimum inside X. Let M be a minimizer. For all i, we define $S_i = \{A \in T_i | \pi_i(A) - M_i(A) = F(M)\}$. In terms of the sets $\{S_i\}_i$ we can define an alternative reduced form π' as follows. For all i, $A \in T_i$: if $A \in S_i$, then set $\pi'_i(A) = \pi_i(A)$; otherwise, set $\pi'_i(A) = \max_{B \in S_i | \pi_i(A) \geq \pi_i(B)} \{\pi(B)\}$, unless $\{B \in S_i | \pi_i(A) \geq \pi_i(B)\}$ is empty in which case set $\pi'_i(A)$ equal to 0. With these changes, the proof is truly identical to that of Lemma 1, and we avoid repeating the steps for brevity. □

Corollary 2. *Every feasible reduced form π for independent bidders and a single item can be exactly implemented as a distribution over strict, partially-ordered w.r.t. π hierarchical mechanisms.*

Proof. The proof is identical to that of Corollary 1 after making the same modifications going from Lemma 1 to Lemma 2. □

We proceed to prove an analog of Proposition 2 in this setting. We need a definition. For all i, let $>^i := A_{i,1} >^i A_{i,2} >^i \ldots >^i A_{i,k_i}$ be a total ordering of the set $T_i := \{A_{i,1}, \ldots, A_{i,k_i}\}$ of bidder i's types. We say that a reduced form π respects $>^i$ iff $\pi_i(A_{i,1}) \geq \pi_i(A_{i,2}) \geq \ldots \geq \pi_i(A_{i,k_i})$. We also say that an hierarchical mechanism H is partially-ordered with respect to $>^i$ iff $H(A_{i,1}, i) \leq H(A_{i,2}, i) \leq \ldots \leq H(A_{i,k_i}, i)$. [4] With respect to these definitions we show the following proposition.

Proposition 4. *For all i, let $>^i := A_{i,1} >^i A_{i,2} >^i \ldots >^i A_{i,k_i}$ be a total ordering of the set $T_i := \{A_{i,1}, \ldots, A_{i,k_i}\}$ of bidder i's types. The set of feasible reduced forms that respect $>^1, \ldots, >^m$ is a $(\sum_i k_i)$-dimensional polytope whose corners are exactly the strict, partially-ordered w.r.t. $>^1, \ldots, >^m$ hierarchical mechanisms.*

Proof. We know from [Bor07, CKM11] that a reduced form π respects $>^1, \ldots, >^m$ and is feasible iff

$$\pi_i(A_{i,j}) \geq \pi_i(A_{i,j+1}) \qquad \forall i \in [m], j \in [k_i] \tag{3.6}$$

$$\sum_i \sum_{j \leq x_i} \Pr[t_i = A_{i,j}] \pi_i(A_{i,j}) \leq 1 - \prod_i \left(1 - \sum_{j \leq x_i} \Pr[t_i = A_{i,j}]\right) \quad \forall x_1 \in [k_1], \ldots, x_m \in [k_m] \tag{3.7}$$

where for notational convenience we denote $\pi_i(A_{i,k_i+1}) = 0$. In fact, to make our lives easier in a later proof, we will actually replace (3.7) with:

$$\sum_i \sum_{j \leq x_i} \Pr[t_i = A_{i,j}] \pi_i(A_{i,j}) \leq 1 - \prod_i \left(1 - \sum_{j \leq x_i} \Pr[t_i = A_{i,j}]\right) \forall x_1 \in [k_1 - 1], \ldots, x_m \in [k_m - 1] \tag{3.8}$$

$$\sum_i \sum_{j \leq k_i} \Pr[t_i = A_{i,j}] \pi_i(A_{i,j}) \leq 1 \tag{3.9}$$

[4] Notice that this definition of partially-ordered hierarchical mechanism (with respect to $\{>^i\}_i$) is similar to its counterpart in the main body (with respect to π), but different. Being partially-ordered with respect to π certainly imposes the same constraints as being partially-ordered with respect to any $\{>^i\}_i$ that π respects. The difference is that being partially-ordered with respect to π may also impose some equality constraints, if $\pi_i(A) = \pi_i(B)$ for types $A \neq B$.

In the above above replacement, we are basically observing that if (3.9) holds, then so does (3.7) for any case where at least one i has $x_i = k_i$. In addition, (3.8) covers all other cases. We have hence shown that the set of feasible reduced forms that respect $>^1, \ldots, >^m$ is a $\sum_i k_i$-dimensional polytope.

We proceed to show that any strict, partially-ordered w.r.t. $>^1, \ldots, >^m$ hierarchical mechanism H whose reduced-form is π is a corner. For convenience in the proof assume that $T_i \cap T_k = \emptyset$, for all $i \neq k$. This is easy to achieve by having every type of bidder i carry the name of bidder i, for all i. Let now $y = \min_x \Pr[\exists i, H(t_i, i) \leq x] = 1$, i.e. the minimum y so that with probability 1 some bidder i will report a type $A_{i,j}$ such that $H(A_{i,j}, i) \leq y$. In terms of this y we define y^* as follows: $y^* := y$, if $y \neq$ LOSE (case 1), and $y^* := y - 1$, if $y =$ LOSE (case 2). We observe then that a type $A_{i,j} \in T_i$ with $H(A_{i,j}, i) > y^*$ cannot possibly win the item, as we are either guaranteed to see some bidder whose type lies on a higher level of the hierarchy (case 1) or the type is mapped to LOSE and is hence given no item (case 2). For all such $i, A_{i,j} \in T_i$, we therefore have $\pi_i(A_{i,j}) = \pi_i(A_{i,j+1}) = \cdots = \pi_i(A_{i,k_i+1}) = 0$.

We say that a set of types $S = \cup_i \{A_{i,1}, \ldots, A_{i,x_i}\}$ for some $x_1, \ldots, x_m$ is *near-constricting*, if the corresponding Inequality (3.7) is tight for $x_1, \ldots, x_m$. Then, for any $i, A_{i,j} \in T_i$ with $H(A_{i,j}, i) \leq y^*$, we know that $H^{-1}([H(A_{i,j}, i)])$ is a near-constricting set, because the item is always awarded to a type in $H^{-1}([H(A_{i,j}, i)])$ whenever at least one type from this set is reported. Moreover, if $H(A_{i,j_1}, i) = \ldots = H(A_{i,j_\ell}, i)$, for some types $A_{i,j_1}, \ldots, A_{i,j_\ell} \in T_i$, then $\pi_i(A_{i,j_1}) = \ldots = \pi_i(A_{i,j_\ell})$. Finally, because H is strict, if some $i, A_{i,j} \in T_i$ satisfy $H(A_{i,j}, i) \leq y^*$, then $H^{-1}(H(A_{i,j}, i)) \cap T_k = \emptyset$, for all $k \neq i$.

Let us now define the following mapping from types to tight inequalities:

- If a type $A_{i,j} \in T_i$ satisfies $H(A_{i,j}, i) > y^*$, then we map $A_{i,j}$ to the constraint $\pi_i(A_{i,j}) = \pi_i(A_{i,j+1})$, i.e. the tightness of inequality $\pi_i(A_{i,j}) \geq \pi_i(A_{i,j+1})$.

- If a type $A_{i,j} \in T_i$ satisfies $H(A_{i,j}, i) \leq y^*$, then:

  - if $H(A_{i,j}, i) = H(A_{i,j+1}, i)$, we map $A_{i,j}$ to the constraint $\pi_i(A_{i,j}) = \pi_i(A_{i,j+1})$, i.e. the tightness of inequality $\pi_i(A_{i,j}) \geq \pi_i(A_{i,j+1})$;

– otherwise, we map $A_{i,j}$ to the tightness of Inequality (3.7) for the set of types $H^{-1}([H(A_{i,j}, i)])$.

The above discussion ensures that our mapping is injective. Hence π makes at least $\sum_i k_i$ of the inequalities defining our polytope tight. And it is not hard to see that there is a unique feasible reduced form making these inequalities tight. So π is a corner of the polytope. Thus, every strict, partially-ordered w.r.t. $>^1, \ldots, >^m$ hierarchical mechanism is a corner of the polytope.

We now make the same observation as in Proposition 2 to argue that there are no other corners. Corollary 2 implies that every point in the polytope can be written as a convex combination of strict, partially-ordered w.r.t. $>^1, \ldots, >^m$ hierarchical mechanisms, all of which are corners of the polytope. As no corner of the polytope can be written as a convex combination of *other* corners of the polytope, there must not be any other corners. □

And now we are ready to prove Theorems 7 and 8.

Proof of Theorem 7: Using a similar argument as in the proof of Theorem 5 we can assume without loss of generality that, for all i, π assigns a distinct probability to every type in T_i. (Otherwise we can define a similar TypeMerge operation, like the one defined in the proof of Theorem 5, whereby types in T_i that receive the same π value are merged into super-types.) Under this assumption, Proposition 4 implies that π lies inside a $(\sum_i |T_i|)$-dimensional polytope, P, whose corners are the strict, partially-ordered w.r.t. π hierarchical mechanisms. By Carathéodory's Theorem, every point in the polytope can be written as a convex combination of at most $(\sum_i |T_i|) + 1$ corners. As a convex combination of corners is exactly a distribution over strict, partially-ordered w.r.t. π hierarchical mechanisms, this proves the theorem. □

Proof of Theorem 8: The first part of the theorem follows immediately as a corollary of Theorem 6.

We now need to describe how to efficiently find a mechanism implementing a reduced form $\tilde{\pi}$ that is feasible. Using a similar argument as in the proof of Theorem 4 we can assume without loss of generality that, for all i, $\tilde{\pi}$ assigns a distinct probability

to every type in T_i. (Otherwise we can merge types in T_i that receive the same $\tilde{\pi}$ value into super-types, like we did in the proof of Theorem 4, then run the procedure outlined below, and finally un-merge types.) Under this assumption, Proposition 4 implies that $\tilde{\pi}$ lies inside a $(\sum_i |T_i|)$-dimensional polytope, P, whose corners are the strict, partially-ordered w.r.t. $\tilde{\pi}$ hierarchical mechanisms. Therefore we can directly apply Theorem 2 of Section 2.8 to write $\tilde{\pi}$ as a convex combination of such hierarchical mechanisms, as long as we can describe the boundary oracle BO, corner oracle CO and separation oracle SO that are needed for Theorem 2. BO is trivial to implement, as we just have to include in the set of halfspaces defining the boundary of P those inequalities described in the proof of Proposition 4. In particular, for convenience in the remaining of the proof, we include the inequalities of the form (3.6), (3.8) and (3.9). For CO, on input B, we first check that every hyperplane $h \in B$ satisfies $BO(h) =$ yes. If not, output no. Otherwise, we need to check if $\bigcap_{h \in B} h$ contains a corner of P. We know that the corners of P are exactly the strict, partially-ordered w.r.t. $\tilde{\pi}$ hierarchical mechanisms. To check if there is such a corner in $\bigcap_{h \in B} h$ we need to do some work.

First, let us use the same notation as in Proposition 4, denoting $T_i = \{A_{i,1}, \ldots, A_{i,k_i}\}$, where $k_i = |T_i|$ and $\tilde{\pi}_i(A_{i,1}) > \ldots > \tilde{\pi}_i(A_{i,k_i})$. Also, let us call a set of types S near-constricting if either $S = \cup_i \{A_{i,1}, \ldots, A_{i,x_i}\}$ for some $x_1 \leq k_1 - 1, \ldots, x_m \leq k_m - 1$ and Inequality (3.8) is tight for $x_1, \ldots, x_m$, or if $S = \cup_i T_i$ and Inequality (3.9) is tight.

Now given a set B of hyperplanes, if B contains a near-constricting set hyperplane for the sets of types $S_1, \ldots, S_k$, we check first whether these sets satisfy $S_1 \subset S_2 \ldots \subset S_k$ (possibly after renaming). If not, then there are some two near-constricting sets S_i, S_j with $A \in S_i - S_j$, $B \in S_j - S_i$ for some types $A \neq B$. Because S_i and S_j are different than $\cup_i T_i$ and they are near-constricting they must be of the form making Inequality (3.8) tight. Hence, both S_i and S_j miss at least one type of every bidder, so that the right-hand side of the inequality for S_i must be < 1 and similarly the right-hand side of the inequality for S_j must be < 1. In addition, we cannot have A and B be types of the same bidder, as we only consider near-constricting sets that respect

the partial ordering within every bidder's type-set. Therefore, A and B belong to different bidders, and because the probability of seeing a type in S_i is < 1 (and the same holds for S_j), there is a non-zero probability that A and B are both present, but no other type of S_i or S_j is present. Then the near-constricting set equation for set S_i requires that we must give the item to A, [5] and the near-constricting set equation for S_j requires that we must give the item to B, which is impossible. So if we do not have $S_1 \subset S_2 \ldots \subset S_k$, the hyperplanes do not intersect in a feasible mechanism, and therefore CO should output no.

Otherwise, CO looks at the other hyperplanes (of the form $\pi_i(A_{i,j}) = \pi_i(A_{i,j+1})$) that belong to B, and chooses an arbitrary strict, partially-ordered w.r.t. $\tilde{\pi}$ hierarchical mechanism that satisfies the following constraints:

1. $H(A,i) < H(A',i')$ for all $(A,i) \in S_j, (A',i') \in S_{j+1} - S_j$, for all j.
2. $H(A,i) < H(A',i')$ for all $(A,i) \in S_k, (A',i') \notin S_k$.
3. For all $i, A_{i,j} \in T_i$:
   (a) if the hyperplanes $\pi_i(A_{i,j}) = \pi_i(A_{i,j+1})$, $\pi_i(A_{i,j+1}) = \pi_i(A_{i,j+2})$, …, $\pi_i(A_{i,k_i}) = 0 (= \pi_i(A_{i,k_i+1}))$ are all in B, then $H(A_{i,j}, i) = \text{LOSE}$;
   (b) otherwise, if $\pi_i(A_{i,j}) = \pi_i(A_{i,j+1})$ is in B, then either $H(A_{i,j}) = H(A_{i,j+1}) = \text{LOSE}$ or $H(A_{i,j}) \geq H(A_{i,j+1}) - 1$.

We claim that an H satisfying the above constraints exists if and only if $\bigcap_{h \in B} h \cap P \neq \emptyset$. By Proposition 4, we know that if there is a corner π in $\bigcap_{h \in B} h \cap P$, there is a strict, partially-ordered w.r.t. $\tilde{\pi}$ hierarchical mechanism H that implements it. Without loss of generality we can make two simplifying assumptions about H: (i) For all $t \neq \text{LOSE}$, $|H^{-1}(t)| \leq 1$. Indeed, suppose that $|H^{-1}(t)| > 1$ for some $t \neq \text{LOSE}$. Then because H is strict, location t of the hierarchy defined by H only contains types belonging to the same bidder i. And because H is partially ordered w.r.t. $\tilde{\pi}$ these types are consecutive in T_i. So we can change H to "expand" cell t of the hierarchy into

[5] Recall that a way to interpret a near-constricting set equation for set S is the following: whenever at least one bidder reports a type in S to the mechanism, the mechanism gives the item to a bidder who reported a type in S, with probability 1.

consecutive cells containing a single type each in the right order. This has no effect in the mechanism defined by H. (ii) If H awards the item to bidder i of type $A_{i,j}$ with probability 0, we can assume that $H(A_{i,j}, i) = \text{LOSE}$. Now given (i) and the nature of $S_1, \ldots, S_k$, [6] for H to have the sets $S_1, \ldots, S_k$ near-constricting it must satisfy the first two constraints above. Indeed for these constraints not to be satisfied there must exist ℓ, $i, A_{i,j}$, $i', A_{i',j'}$, $i \neq i'$ such that $H(A_{i,j}, i) < H(A_{i',j'}, i')$ and $A_{i,j} \notin S_\ell$ while $A_{i',j'} \in S_\ell$. But then it must be $\pi(A_{i,j}) > 0$ (because of assumption (ii)), so there is a positive probability that bidder i of type $A_{i,j}$ is the highest in the hierarchy when every bidder except for i' has declared a type. Still, given that $H(A_{i,j}, i) < H(A_{i',j'}, i')$, even if i' declares $A_{i',j'}$, i will get the item, contradicting that S_ℓ is near-constricting. We argue next that H also needs to satisfy the third constraint above. That constraint 3(a) is satisfied follows immediately from assumption (ii). We argue next that 3(b) needs to be satisfied. Indeed, suppose there exist $A_{i,j}, A_{i,j+1} \in T_i$ and $A_{i',j'} \in T_{i'}$ such that $\pi_i(A_{i,j}) = \pi_i(A_{i,j+1}) > 0$ and $H(A_{i,j}, i) < H(A_{i',j'}, i') < H(A_{i,j+1}, i)$. As $\pi_i(A_{i,j+1}) > 0$, it follows that $H(A_{i,j+1}, i) < \text{LOSE}$, so $H(A_{i',j'}, i') < \text{LOSE}$, which implies that $\pi_{i'}(A_{i',j'}) > 0$ (otherwise it would be that $H(A_{i',j'}, i') = \text{LOSE}$ given (ii)). Now, because bidder i' wins the item with non-zero probability as type $A_{i',j'}$, there is a non-zero probability that $A_{i',j'}$ is on the highest level of the hierarchy after sampling from all bidders except for i. In this case, i will win the item by reporting $A_{i,j}$, and lose by reporting $A_{i,j+1}$. In all other cases, i is at least as likely to win the item by reporting $A_{i,j}$ as $A_{i,j+1}$, and therefore we see that bidder i gets the item strictly more often when reporting $A_{i,j}$ than $A_{i,j+1}$, violating the constraint $\pi_i(A_{i,j}) = \pi_i(A_{i,j+1})$. So if there is a corner π in $\bigcap_{h \in B} h \cap P$ it can be implemented by a strict, partially-ordered w.r.t. $\tilde{\pi}$ hierarchical mechanism H satisfying the above constraints. The other direction of our claim is even simpler. If a strict, partially-ordered w.r.t. $\tilde{\pi}$ hierarchical mechanism H satisfies all constraints above, then its induced reduced-form π will immediately satisfy all equalities in B.

Hence to implement CO one just needs to check if there is a strict, partially-

[6] in particular, the fact that the sets $S_1, \ldots, S_k$ respect the ordering of the type sets as follows: For all ℓ, $A_{i,j} \in S_\ell$ implies $A_{i,j'} \in S_\ell$, for all $j' \leq j$.

ordered w.r.t. $\tilde{\pi}$ hierarchical mechanism H satisfying the four constraints above. This task is easy to do efficiently. If a mechanism is found it satisfies all equalities in B and it is a corner of the polytope by Proposition 4.

SO is also simple to implement. On input $\vec{\pi}$, we first check that Inequalities (3.6) are satisfied (i.e. that $\vec{\pi}$ respects the total orderings on the bidders' type-spaces induced by $\tilde{\pi}$). Then we use the separation oracle provided by Theorem 6 to verify that $\vec{\pi}$ is feasible. As all three oracles BO, CO, SO run in polynomial time, we can apply Theorem 2 to write $\tilde{\pi}$ as a convex combination of at most $\sum_i |T_i| + 1$ corners, which is exactly a distribution over at most $\sum_i |T_i| + 1$ strict, partially-ordered hierarchical mechanisms in polynomial time. $\square$

3.4 Implementation of Single-item Reduced Forms via Hierarchical Mechanisms

Here we provide the proof of Theorem 9.

Proof of Theorem 9: Let σ be a total ordering over all possible types, $\sigma : \cup_i (T_i \times \{i\}) \to [\sum_i |T_i|]$. Define the unhappiness $F_\sigma(M)$ of a distribution over σ-ordered hierarchical mechanisms, M, as follows:

$$F_\sigma(M) = \max_{i, A \in T_i} (\pi_i(A) - M_i(A)).$$

As we argued formally in earlier sections F_σ can be viewed as a continuous function over a compact set. Hence it achieves its minimum inside the set. Let then $M^\sigma \in \text{argmin}_M F_\sigma(M)$ (where the minimization is over all distributions over σ-ordered hierarchical mechanisms) and define the set S_σ to be the set of maximally unhappy types under M^σ; formally, $S_\sigma = \text{argmax}_{i,A}\{\pi_i(A) - M_i^\sigma(A)\}$. If for some σ there are several minimizers M^σ, choose one that minimizes $|S_\sigma|$. Now, let MO be the set of the orderings σ that minimize $F_\sigma(M^\sigma)$. Further refine MO to only contain σ's minimizing $|S_\sigma|$. Formally, we first set $MO = \text{argmin}_\sigma\{F_\sigma(M^\sigma)\}$ and then refine MO as $MO_{\text{new}} = \text{argmin}_{\sigma \in MO}\{|S_\sigma|\}$. We drop the subscript "new" for the rest of the proof.

From now on, we call a type (A, i) *happy* if $M_i(A) \geq \pi_i(A)$, otherwise we call (A, i) *unhappy*. Intuitively, here is what we have already done: For every ordering σ, we have found a distribution over σ-ordered hierarchical mechanisms M^σ that minimizes the maximal unhappiness and subject to this, the number of maximally unhappy types. We then choose from these (σ, M^σ) pairs those that minimize the maximal unhappiness, and subject to this, the number of maximally unhappy types. We have made these definitions because we want to eventually show that there is an ordering σ, such that $F_\sigma(M^\sigma) \leq 0$, and it is natural to start with the ordering that is "closest" to satisfying this property. We are one step away from completing the proof. What we will show next is that, if $\tau \in MO$ does not make every type happy,

then we can find some other ordering τ', such that $F_{\tau'}(M^{\tau'}) = F_\tau(M^\tau)$, $|S_{\tau'}| = |S_\tau|$, and $S_{\tau'} = \{\tau^{-1}(1), \ldots, \tau^{-1}(|S_{\tau'}|)\}$. In other words, only the top $|S_{\tau'}|$ types in τ are maximally unhappy. From here, we will show that because $\tau' \in MO$, that $S_{\tau'}$ is a constricting set and get a contradiction.

First, if the maximally unhappy types in S_τ are not the top $|S_\tau|$ ones, let i be the smallest i such that $\tau^{-1}(i+1) \in S_\tau$ but $\tau^{-1}(i) \notin S_\tau$. We proceed to show that by changing either the distribution M or the ordering τ, we can always move $\tau^{-1}(i)$ into S_τ and $\tau^{-1}(i+1)$ out without changing $|S_\tau|$ or the value $F_\tau(M)$. Then by repeating this procedure iteratively, we will get the τ' we want.

Before we describe the procedure, we introduce some terminology. We say there is a *cut* between $\tau^{-1}(i)$ and $\tau^{-1}(i+1)$ in a fixed τ-ordered hierarchical mechanism H if $H(\tau^{-1}(i)) < H(\tau^{-1}(i+1))$, i.e. if $\tau^{-1}(i)$ and $\tau^{-1}(i+1)$ are on different levels of the hierarchy. For the remainder of the proof, we will let l be the level of $\tau^{-1}(i)$ ($H(\tau^{-1}(i))$). When we talk about adding or removing a cut below i, we mean increasing or decreasing $H(\tau^{-1}(j))$ by 1 for all $j > i$. We now proceed with a case analysis, for fixed $\tau^{-1}(i) \notin S_\tau$, $\tau^{-1}(i+1) \in S_\tau$. We let $(A, j) = \tau^{-1}(i)$ and $(B, k) = \tau^{-1}(i+1)$.

- **Case 1:** $j = k$.

  Since τ is a linear extension of the bidder's own ordering, then $\pi_j(A) \geq \pi_j(B)$, but we know that

  $$\pi_j(A) - M_j^\tau(A) < \pi_j(B) - M_j^\tau(B),$$

  thus $M_j^\tau(A) > M_j^\tau(B) \geq 0$. Because A and B are types for the same bidder j, when A and B are in the same level, they get the item with equal probability. Therefore, there must exist some $H \in \text{supp}(M^\tau)$ with a cut below A, and in which A gets the item with non-zero probability. We modify M^τ by modifying the mechanisms H in its support as follows.

  Let H be a hierarchical mechanism in the support of M^τ. If there is no cut below A, we do nothing. If all of the types on level l *and* level $l+1$ are from bidder j, we remove the cut below A. This does not affect $H_q(C)$ (the probability that

(C, q) gets the item under H) for any $q, C \in T_q$, because it was impossible for two types in the combined level to show up together anyway. As we have not changed $H_q(C)$ for any q, C in the mechanisms we have touched so far, yet none of these mechanisms has a cut between levels l and $l+1$, there must still be some $H \in \text{supp}(M^\tau)$ with a cut below A and in which A gets the item with non-zero probability (otherwise it couldn't be that $M_j^\tau(A) > M_j^\tau(B) \geq 0$). For such an H, there is at least one type not from bidder j in level l or $l+1$. We distinguish two sub-cases:

- Every bidder has at least one type in level $l+1$ or larger (in other words, every type in level $l+1$ wins the item with non-zero probability). Consider now moving the cut from below i to below $i-1$. Clearly, A will be less happy if we do this. Every type not from bidder j in l will be strictly happier, as now they do not have to share the item with A. Every type not from bidder j in $l+1$ will be strictly happier, as they now get to share the item with A. It is also not hard to see that all types $\neq A$ from bidder j in level l and $l+1$ are not affected by this change, as they never share the item with A in either case. So in particular B is unaffected. Consider instead moving the cut from below i to below $i+1$. Then B is happier, every type not from bidder j in $l+1$ is less happy than before (as they now don't get to share with B), every type not from bidder j in l is also less happy than before (because now they have to share with B), and all types $\neq B$ from bidder j in level l and $l+1$ are not affected by the change (as they never share the item with B in either case). To summarize, we have argued that, when we move the cut to below $i+1$, B becomes strictly happier, and every type that becomes less happy by this change becomes strictly happier if we move the cut to below $i-1$ instead. Also, B is unaffected by moving the cut to $i-1$. So with a tiny probability ϵ, move the cut from below i to below $i-1$, whenever H is sampled from M^τ. This makes all of the types not from bidder j in level

l or $l+1$ strictly happier. With a tinier probability δ, move the cut from below i to below $i+1$, whenever H is sampled from M^τ. Choose ϵ to be small enough that we don't make A maximally unhappy, and choose δ to be small enough so that we don't make any types besides A less happy than they were in H. Then we have strictly increased the happiness of B without making A maximally unhappy, or decreasing the happiness of any other types. Therefore, we have reduced $|S_\tau|$, a contradiction.

- If there is a bidder j' whose types are all in levels $1, \ldots, l$ (call such bidders *high*), then no type in level $l+1$ can possibly win the item. We also know that: every high bidder has at least one type in level l by our choice of H (otherwise A would get the item with probability 0); and all high bidders are different than j, since B is in level $l+1$. Now we can basically use the same argument as above. The only difference is that when we move the cut to below $i-1$ or the cut to below $i+1$, types in level $l+1$ that are different than B will remain unaffected (i.e. the affected types different from B are only those in level l). But since every high bidder has a type in level l, B will be unaffected in the first case but strictly happier in the second, and it is still the case that every type who is made unhappier by moving the cut to below $i+1$ is made strictly happier by moving the cut to below $i-1$. So we can carry over the same proof as above, and get a contradiction.

Therefore, it can not be the case that $j = k$.

- **Case 2:** $j \neq k$ and there is never a cut below A.

  This case is easy. If we switch (A, j) and (B, k) in τ, then the set S_τ is exactly the same, and the distribution M^τ is exactly the same. However, we have now relabeled the types in S_τ so that $\tau^{-1}(i) \in S_\tau$ and $\tau^{-1}(i+1) \notin S_\tau$.

- **Case 3:** $j \neq k$ and there is sometimes a cut below A.

  Pick a mechanism H in the support of M^τ that has a cut between A and B

and in which A gets the item with positive probability. (If such a mechanism doesn't exist we can remove the cut between i and $i+1$ in all mechanisms in the support without changing the allocation probabilities and return to Case 2). Let now $i^* = \max_{i'<i}\{i' | \tau^{-1}(i') \in S_\tau\}$. By our choice of i (specifically, that it is the smallest i such that $\tau^{-1}(i+1) \in S_\tau$ but $\tau^{-1}(i) \notin S_\tau$), we see that $\tau^{-1}(i') \in S_\tau$ for all $i' \leq i^*$, and $\tau^{-1}(i') \notin S_\tau$ for all $i^* < i' \leq i$. There are again two sub-cases:

- $H(\tau^{-1}(i^*)) < l$. By our choice of i^*, this means that everyone in level l is *not* maximally unhappy. By our choice of H, everyone in level l receives the item with non-zero probability, so there is at least one type from each bidder in level l or larger. If we pick a tiny ϵ, and with probability ϵ remove the cut from below i (whenever H is sampled from M^τ), then everyone in level $l+1$ is happier, everyone in level l is unhappier, and everyone else is unaffected. In particular, B will be strictly happier with this change, as he now gets to share with A (and possibly others). If we choose a sufficiently small ϵ, no one in level l will be made maximally unhappy, and (B, k) will be removed from S_τ, a contradiction.
- $H(\tau^{-1}(i^*)) = l$. In this case, introduce a cut below i^* with some probability ϵ whenever H is sampled from M^τ. The only types who may become happier by this change are those in level l with $\tau(C, q) \leq i^*$. The only types who may become unhappier by this change are those in level l with $\tau(C, q) > i^*$. Everyone else is unaffected by this change. But, if we can make any type happier, then we can choose ϵ small enough, so that we remove this type from S_τ (this type must be in S_τ as all types in level l with $\tau(C, q) \leq i^*$ are) without making any new type maximally unhappy (as all types that can possibly become unhappier with this change are *not* in S_τ). Again, we obtain a contradiction because this would decrease $|S_\tau|$ without increasing $F_\tau(M^\tau)$. Thus, this change cannot make anyone happier, and therefore cannot make anyone unhappier. So we may modify

M^σ by introducing a cut below i^* with probability 1 whenever M^τ samples H, thereby removing H from the support of M^τ (without making anyone happier or unhappier) and replacing it with H' satisfying: $H'(\tau^{-1}(i^*)) < H'(\tau^{-1}(i)) < H'(\tau^{-1}(i+1))$ and H' awards the item to $\tau^{-1}(i)$ with non-zero probability. After this modification, we may return to the previous sub-case to obtain a contradiction.

Hence, it can not be the case that $j \neq k$ with sometimes a cut below A.

At the end of all three cases, we see that if we ever have $\tau^{-1}(i) \notin S_\tau$ and $\tau^{-1}(i+1) \in S_\tau$, then these types must belong to different bidders, and no mechanism in the support of M^τ ever places a cut between these types. Hence, we can simply swap these types in τ (as we described in Case 2 above), and we do that repeatedly until we have $S_\tau = \{\tau^{-1}(1), \ldots, \tau^{-1}(|S_\tau|)\}$. Once such a τ has been found, let $k = |S_\tau|$. Now consider a mechanism in the support of M^τ that has no cut below k, and consider putting a cut there with some tiny probability ϵ whenever this mechanism is sampled. The only effect this *might* have is that when the item went to a type outside S_τ, it now goes with some probability to a type inside S_τ. Therefore, if anyone gets happier, it is someone in S_τ. However, if we make anyone in S_τ happier and choose ϵ small enough so that we don't make anyone outside of S_τ maximally unhappy, we decrease $|S_\tau|$, getting a contradiction. Therefore, putting a cut below k cannot possibly make anyone happier, and therefore cannot make anyone unhappier. So we may w.l.o.g. assume that there is a cut below k in all mechanisms in the support of M^τ. But now we get that the item always goes to someone in S_τ whenever someone in S_τ shows up, yet everyone in this set is unhappy. Therefore, S_τ is a constricting set, certifying that the given π is infeasible.

Putting everything together, we have shown that if there is no σ with $F_\sigma(M^\sigma) \leq 0$ then the reduced form is infeasible. So there must be some σ with $F_\sigma(M^\sigma) \leq 0$, and such an M^σ implements the reduced form by sampling only σ-ordered hierarchical mechanisms, completing the proof. $\square$

Chapter 4

Feasibility of General Reduced Forms

As we have seen in Chapter 3, checking feasibility for reduced form auctions is a non-trivial task even with only a single item. In this chapter, we study the feasibility of general reduced form auctions with arbitrary feasibility constraint. Surprisingly, we manage to find a general approach to solve it.

We first show a characterization result that every feasible auction can be implemented as a distribution over *virtual VCG allocation rules*. A virtual VCG allocation rule has the following simple form: Every bidder's type t_i is transformed into a virtual type $f_i(t_i)$, via a bidder-specific function. Then, the allocation maximizing virtual welfare is chosen. We generalize this result to arbitrarily correlated bidders, introducing the notion of a *second-order* VCG allocation rule. Next, we give two algorithmic results on reduced form auctions in settings with arbitrary feasibility and demand constraints. First, we provide a separation oracle for determining feasibility of a reduced form auction. Second, we provide a geometric algorithm to decompose any feasible reduced form into a distribution over virtual VCG allocation rules. In addition, we show how to efficiently execute both algorithms given only black box access to an implementation of the VCG allocation rule.

Section 4.1 provides the characterization result. In Section 4.2, we present an exact but inefficiently implementation of the separation oracle and decomposition

algorithm using the characterization result. In Section 4.3, we briefly describe how to use a novel sampling technique to make both algorithms efficient, and give full details in Section 4.4. In Section 4.5, we present the characterization result for correlated bidders.

4.1 Characterization of General Feasible Reduced Forms

In this section, we provide our characterization result, showing that every feasible reduced form can be implemented as a distribution over virtual VCG allocation rules. In the following definition, $VCG_{\mathcal{F}}$ denotes the allocation rule of VCG with feasibility constraints $\mathcal{F}$. That is, on input $\vec{v} = (\vec{v}_1, \ldots, \vec{v}_m)$, $VCG_{\mathcal{F}}$ outputs the allocation that VCG selects when the reported types are $\vec{v}$.

Definition 7. *A **virtual VCG** allocation rule is defined by a collection of weight functions, $f_i : T_i \to \mathbb{R}^n$. f_i maps a type of bidder i to a* virtual type *of bidder i. On any type profile $\vec{v}$, the virtual VCG allocation rule with functions $\{f_i\}_{i\in[m]}$ runs $VCG_{\mathcal{F}}$ on input $(f_1(\vec{v}_1), \ldots, f_m(\vec{v}_m))$.[1] $VVCG_{\mathcal{F}}(\{f_i\}_{i\in[m]})$ denotes the virtual VCG allocation rule with feasibility constraints $\mathcal{F}$ and weight functions $\{f_i\}_{i\in[m]}$.*

In other words, a virtual VCG allocation rule is simply a VCG allocation rule, but maximizing virtual welfare instead of true welfare. It will be convenient to introduce the following notation, viewing the weight functions as a (scaled) $n\sum_{i=1}^{m}|T_i|$-dimensional vector. Below, f_{ij} denotes the j^{th} component of f_i.

Definition 8. *Let $\vec{w} \in \mathbb{R}^{n\sum_{i=1}^{m}|T_i|}$. Define f_i so that $f_{ij}(A) = \frac{w_{ij}(A)}{\Pr[t_i=A]}$. Then $VVCG_{\mathcal{F}}(\vec{w})$ is the virtual VCG allocation rule $VVCG_{\mathcal{F}}(\{f_i\}_{i\in[m]})$.*

It is easy to see that every virtual VCG allocation rule can be defined using the notation of Definition 8 by simply setting $w_{ij}(A) = f_{ij}(A) \cdot \Pr[t_i = A]$. We scale

[1] If there are multiple VCG allocations, break ties arbitrarily, but consistently. A consistent lexicographic tie-breaking rule is discussed in Section 4.1. For concreteness, the reader can use this rule for all results of this section.

the weights this way only for notational convenience (which first becomes useful in Lemma 3). We say that a virtual VCG allocation rule is simple iff, for all $\vec{v}_1, \ldots, \vec{v}_m$, $VCG_{\mathcal{F}}(f_1(\vec{v}_1), \ldots, f_m(\vec{v}_m))$ has a unique max-weight allocation. We now state the main theorem of this section, which completely characterizes all feasible reduced forms.

Theorem 10. *Let $\mathcal{F}$ be any set system of feasibility constraints, and $\mathcal{D}$ be any (possibly correlated) distribution over bidder types with finite support. Then every feasible reduced form (with respect to $\mathcal{F}$ and $\mathcal{D}$) can be implemented as a distribution over at most $n\sum_{i=1}^{m}|T_i|+1$ simple virtual VCG allocation rules.*

Before outlining the proof, we provide a brief example illustrating the content of Theorem 10. We are *not* claiming that every feasible allocation rule can be implemented as a distribution over virtual VCG allocation rules. This is not true. What we are claiming is that every feasible allocation rule has the same reduced form as some distribution over virtual VCG allocation rules. Consider a scenario with a single item and two bidders each with two types, A and B that are sampled independently and uniformly at random. If M is the allocation rule that awards bidder 1 the item when the types match, and bidder 2 the item when they don't, then M cannot be implemented as a distribution over simple virtual VCG allocation rules. Because bidder 1 gets the item when both types match, we must always have $w_{11}(A) > w_{21}(A)$ and $w_{11}(B) > w_{21}(B)$. Similarly, because bidder 2 gets the item when the types don't match we must have $w_{21}(A) > w_{11}(B)$ and $w_{21}(B) > w_{11}(A)$. Clearly, no weights can simultaneously satisfy all four inequalities. However, there is a distribution over simple virtual VCG allocation rules with the same reduced form.[2] The proof of Theorem 10 begins with a simple observation and proposition.

Observation 3. *An allocation rule is feasible if and only if it is a distribution over feasible deterministic allocation rules.*

[2]Specifically, the reduced form of M is $\frac{1}{2}\cdot\vec{1}$. If we define $w_{11}^{(1)}(A) = w_{11}^{(1)}(B) = 1$, $w_{21}^{(1)}(A) = w_{21}^{(1)}(B) = 0$, and $w_{11}^{(2)}(A) = w_{11}^{(2)}(B) = 0$, $w_{21}^{(2)}(A) = w_{21}^{(2)}(B) = 1$, then the allocation rule that chooses uniformly at random between $VVCG_{\mathcal{F}}(\vec{w}^{(1)})$ and $VVCG_{\mathcal{F}}(\vec{w}^{(2)})$ also has reduced form $\frac{1}{2}\cdot\vec{1}$.

Proof. For any feasible allocation rule M, and any type profile $\vec{v}$, the (possibly randomized) allocation $M(\vec{v})$ is a distribution over feasible deterministic allocations. So let $M(\vec{v})$ sample the deterministic allocation $A_i(\vec{v})$ with probability $p_i(\vec{v})$. Then $M(\vec{v})$ can be implemented by uniformly sampling x from $[0, 1]$ and selecting $A_i(\vec{v})$ iff $\sum_{j<i} p_j(\vec{v}) < x \leq \sum_{j \leq i} p_j(\vec{v})$. So for $y \in [0, 1]$ let $M^{(y)}$ denote the deterministic allocation rule that on profile $\vec{v}$ implements the deterministic allocation selected by $M(\vec{v})$ when $x = y$, then M is exactly the allocation rule that samples x uniformly at random from $[0, 1]$ and implements the deterministic allocation rule $M^{(x)}$. So every feasible allocation rule is a distribution over deterministic allocation rules. The other direction is straight-forward: any distribution over feasible deterministic allocation rules is still feasible. □

Proposition 5. *If $|\mathcal{D}|$ is finite, $F(\mathcal{F}, \mathcal{D})$ is a convex polytope.*

Proof. It is clear that there are only finitely many deterministic allocation rules: there are finitely many choices per profile, and finitely many profiles. So consider the set S that contains the reduced form of every deterministic allocation rule that is feasible with respect to $\mathcal{F}$. We claim that $F(\mathcal{F}, \mathcal{D})$ is exactly the convex hull of S. Consider any feasible reduced form $\vec{\pi}$. Then there is some feasible allocation rule M that implements $\vec{\pi}$. By Observation 3, M is a distribution over deterministic allocation rules, sampling M_i with probability p_i. Therefore, if $\vec{\pi}_i$ denotes the reduced form of M_i, we must have $\vec{\pi} = \sum_i p_i \vec{\pi}_i$, so $\vec{\pi}$ is in the convex hull of S. Similarly, if a reduced form $\vec{\pi}$ satisfies $\vec{\pi} = \sum_i p_i \vec{\pi}_i$, where $\vec{\pi}_i$ is the reduced form of a deterministic allocation rule M_i for all i, the allocation rule that selects M_i with probability p_i implements $\vec{\pi}$. So the space of feasible reduced forms is exactly the convex hull of S, which is finite, and hence its convex hull is a polytope. □

Now that we know that $F(\mathcal{F}, \mathcal{D})$ is a convex polytope, we want to look at the extreme points by examining, for any $\vec{w}$, the allocation rule of $F(\mathcal{F}, \mathcal{D})$ whose reduced form maximizes $\vec{\pi} \cdot \vec{w}$. Lemma 3 and Proposition 6 characterize the extreme points of $F(\mathcal{F}, \mathcal{D})$, which allows us to prove Theorem 10.

Lemma 3. *Let $\vec{\pi}$ be the reduced form of $VVCG_{\mathcal{F}}(\vec{w})$ (with an arbitrary tie-breaking rule) when bidders are sampled from $\mathcal{D}$. Then, for all $\vec{\pi}' \in F(\mathcal{F}, \mathcal{D})$, $\vec{\pi} \cdot \vec{w} \geq \vec{\pi}' \cdot \vec{w}$.*

Proof. The proof is straight-forward once we correctly interpret $\vec{\pi} \cdot \vec{w}$. Expanding the dot product and using that $f_{ij}(A) = w_{ij}(A)/\Pr[t_i = A]$, we see that:

$$\begin{aligned}\vec{\pi} \cdot \vec{w} &= \sum_i \sum_j \sum_{A \in T_i} \pi_{ij}(A) w_{ij}(A) \\ &= \sum_i \sum_j \sum_{A \in T_i} \pi_{ij}(A) f_{ij}(A) \Pr[t_i = A].\end{aligned}$$

If the "weight" of awarding item j to bidder i when her reported type is A is $f_{ij}(A)$, then the last line is exactly the expected weight of items awarded by an allocation rule whose reduced form is $\vec{\pi}$. The feasible allocation rule that maximizes the expected weight of items awarded simply chooses a max-weight feasible allocation on every profile. This is exactly what $VVCG_{\mathcal{F}}(\{f_i\}_{i \in [m]})$ does, i.e. exactly what $VVCG_{\mathcal{F}}(\vec{w})$ does. So the reduced form of $VVCG_{\mathcal{F}}(\vec{w})$ exactly maximizes $\vec{x} \cdot \vec{w}$ over all $\vec{x} \in F(\mathcal{F}, \mathcal{D})$. □

Proposition 6. *Every corner of $F(\mathcal{F}, \mathcal{D})$ can be implemented by a simple virtual VCG allocation rule, and the reduced form of any simple virtual VCG allocation rule is a corner of $F(\mathcal{F}, \mathcal{D})$.*

Proof. We first prove that every corner of $F(\mathcal{F}, \mathcal{D})$ can be implemented by a simple virtual VCG allocation rule. From Proposition 5, we know $F(\mathcal{F}, \mathcal{D})$ is a convex polytope. So for every corner $\vec{\pi} \in F(\mathcal{F}, \mathcal{D})$, there is a weight vector $\vec{w}$, such that $\forall\ \vec{\pi}' \in F(\mathcal{F}, \mathcal{D})$ and $\vec{\pi}' \neq \vec{\pi}$,

$$\vec{w} \cdot \vec{\pi} > \vec{w} \cdot \vec{\pi}'.$$

So by Lemma 3, we know that $\vec{\pi}$ must be the reduced form of $VVCG_{\mathcal{F}}(\vec{w})$, as $\vec{\pi}$ maximizes $\vec{x} \cdot \vec{w}$ over all $\vec{x} \in F(\mathcal{F}, \mathcal{D})$. To see that $VVCG_{\mathcal{F}}(\vec{w})$ is simple, assume for contradiction that there is some profile with multiple max-weight feasible allocations. Let B denote the allocation rule that chooses the exact same allocation as $VVCG_{\mathcal{F}}(\vec{w})$ on every other profile, but chooses a different max-weight feasible allocation on this

profile. Let $\vec{\pi}_B$ denote the reduced form of B. By the definition of B, we still have $\vec{\pi}_B \cdot \vec{w} = \vec{\pi} \cdot \vec{w}$. Yet, we also clearly have $\vec{\pi}_B \neq \vec{\pi}$, as they are reduced forms for allocation rules that are identical on all but one profile, where they differ. This contradicts the choice of $\vec{w}$, so $VVCG_{\mathcal{F}}(\vec{w})$ must be simple.

Now we show that the reduced form of any simple virtual VCG allocation rule is a corner of $F(\mathcal{F}, \mathcal{D})$. Let $\vec{\pi}$ be the reduced form of $VVCG_{\mathcal{F}}(\vec{w})$. Then for any other $\vec{\pi}' \in F(\mathcal{F}, \mathcal{D})$, we must have $\vec{\pi} \cdot \vec{w} > \vec{\pi}' \cdot \vec{w}$. Otherwise, let $\vec{\pi}'$ denote a feasible reduced form with $\vec{\pi}' \cdot \vec{w} \geq \vec{\pi} \cdot \vec{w}$, $\vec{\pi}' \neq \vec{\pi}$ and let M' implement $\vec{\pi}'$. Then clearly, there is some profile where the allocation chosen by M' differs from that chosen by $VVCG_{\mathcal{F}}(\vec{w})$ and its weight with respect to $\{f_i\}_{i \in [m]}$ is at least as large as the weight of the allocation chosen by $VVCG_{\mathcal{F}}(\vec{w})$. As $VVCG_{\mathcal{F}}(\vec{w})$ is simple, this is a contradiction. Therefore, $\vec{\pi} \cdot \vec{w} > \vec{\pi}' \cdot \vec{w}$ for all $\vec{\pi}' \neq \vec{\pi} \in F(\mathcal{F}, \mathcal{D})$ and $\vec{\pi}$ is a corner. □

Now we are ready to prove Theorem 10.

Proof of Theorem 10: Since $F(\mathcal{F}, \mathcal{D})$ is a convex polytope (Proposition 5), by Carathéodory's Theorem, we know that every point in $F(\mathcal{F}, \mathcal{D})$ can be written as a convex combination of at most $n \sum_{i=1}^{m} |T_i| + 1$ corners of $F(\mathcal{F}, \mathcal{D})$. By Proposition 6, we know that every corner of $F(\mathcal{F}, \mathcal{D})$ can be implemented by a simple virtual VCG allocation rule. Finally, we observe that if the allocation rules M_i implement $\vec{\pi}_i$, then the allocation rule that samples M_i with probability p_i implements $\sum_i p_i \vec{\pi}_i$. □

We conclude this section by providing necessary and sufficient conditions for feasibility of a reduced form. For the following statement, for any weight vector $\vec{w} \in \mathbb{R}^{n \sum_{i=1}^{m} |T_i|}$, $W_{\mathcal{F}}(\vec{w})$ denotes the total expected weight of items awarded by $VVCG_{\mathcal{F}}(\vec{w})$ (where we assume that the weight of giving item j to bidder i of type A is $f_{ij}(A) = w_{ij}(A)/\Pr[t_i = A]$). The proof of Lemma 3 implies that the tie-breaking rule used in $VVCG_{\mathcal{F}}(\vec{w})$ does not affect the value of $W_{\mathcal{F}}(\vec{w})$, and that no feasible allocation rule can possibly exceed $W_{\mathcal{F}}(\vec{w})$. The content of the next corollary is that this condition is also sufficient.

Corollary 3. *A reduced form $\vec{\pi}$ is feasible (with respect to $\mathcal{F}$ and $\mathcal{D}$) if and only if, for all $\vec{w} \in [-1, 1]^{n \sum_{i=1}^{m} |T_i|}$, $\vec{\pi} \cdot \vec{w} \leq W_{\mathcal{F}}(\vec{w})$.*

Proof. As $F(\mathcal{F}, \mathcal{D})$ is a convex polytope, we know that $\vec{\pi} \in F(\mathcal{F}, \mathcal{D})$ if and only if for all $\vec{w} \in [-1, 1]^{n \sum_{i=1}^{m} |T_i|}$:

$$\vec{\pi} \cdot \vec{w} \leq \max_{\vec{\pi}' \in F(\mathcal{F}, \mathcal{D})} \vec{\pi}' \cdot \vec{w}.$$

By the definition of $VVCG_{\mathcal{F}}(\vec{w})$, the right hand side is exactly $W_{\mathcal{F}}(\vec{w})$. □

Tie-breaking

Here we discuss tie-breaking. This is important in later sections because we will want to argue that any virtual VCG allocation rule we use is simple. Because we only have black-box access to $A_{\mathcal{F}}$, we do not necessarily have any control over the tie-breaking rule used, which could be problematic. Instead, we would like to enforce a particularly simple tie-breaking rule by changing $\vec{w}$ to $\vec{w}'$ such that $VVCG_{\mathcal{F}}(\vec{w}')$ also maximizes $\vec{\pi} \cdot \vec{w}$ over all reduced forms in $F(\mathcal{F}, \mathcal{D})$, and $VVCG_{\mathcal{F}}(\vec{w}')$ is simple. Additionally, we would like the bit complexity of coordinates of $\vec{w}'$ to be polynomial in the bit complexity of coordinates of $\vec{w}$. Lemma 4 formally that a simple lexicographic tie-breaking rule can be implemented in the desired manner.

Lemma 4. *Let $\vec{w}$ be a weight vector whose coordinates are rational numbers of bit complexity ℓ_1, and let ℓ_2 be such that for all $i, A \in T_i$, $\Pr[t_i = A]$ is a rational number of bit complexity ℓ_2. Then the lexicographic tie-breaking rule can be implemented by a simple transformation that turns $\vec{w}$ into $\vec{w}'$ such that $VVCG_{\mathcal{F}}(\vec{w}')$ is simple, $VVCG_{\mathcal{F}}(\vec{w}')$ selects a maximum weight allocation with respect to the scaled weights $\vec{w}$ on every profile, and each coordinate of $\vec{w}'$ is a rational number of bit complexity $n\ell_1 \sum_{i=1}^{m} |T_i| + (n \sum_{i=1}^{m} |T_i| + 1)\ell_2 + mn + n + 1$.*

Proof of Lemma 4: Let $\{f_i\}_i$ denote the weight functions used by $VVCG_{\mathcal{F}}(\vec{w})$ (i.e. $f_{ij}(A) = w_{ij}(A)/\Pr[t_i = A]$), and rewrite each value $f_{ij}(A)$ with a common denominator. Before rewriting, each $f_{ij}(A)$ was a rational number of bit complexity $\ell_1 + \ell_2$. After rewriting, the numerator and denominator of each $f_{ij}(A)$ has at most $b = n(\sum_{i=1}^{m} |T_i|)(\ell_1 + \ell_2)$ bits. Now define new weight functions $\{f'_i\}_i$ such that $f'_{ij}(A)$ is equal to $f_{ij}(A)$ except that $2^{-b-ni-j-1}$ is added to its numerator. In

going from $\{f_i\}_i$ to $\{f'_i\}_i$, the numerator of the weight of any allocation goes up by at most 2^{-b-n-1}, not enough to make an allocation optimal if it was suboptimal. Moreover, notice that the last mn bits of the numerator are in one-to-one correspondence with possible allocations. So even if there were any ties before, there can't be any ties after the transformation, and ties will be broken lexicographically. It is also obvious that we have only added $nm + n + 1$ bits to each numerator of $f_{ij}(A)$, and therefore $w'_{ij}(A) = f'_{ij}(A) \Pr[t_i = A]$ has bit complexity $b + nm + n + 1 + \ell_2 = n\ell_1 \sum_{i=1}^{m} |T_i| + (n \sum_{i=1}^{m} |T_i| + 1)\ell_2 + mn + n + 1$. $\square$

From now on, whenever we use the term $VVCG_{\mathcal{F}}(\vec{w})$, we will implicitly assume that this tie-breaking rule has been applied. Sometimes we will explicitly state so, if we want to get our hands on $\vec{w}'$.

4.2 Algorithms for Reduced Forms

The characterization result of Section 4.1 hinges on the realization that $F(\mathcal{F}, \mathcal{D})$ is a polytope whose corners can be implemented by especially simple allocation rules, namely simple virtual VCG allocation rules. To compute the reduced form of an optimal mechanism, we would like to additionally optimize a linear objective (expected revenue) over $F(\mathcal{F}, \mathcal{D})$, so we need a separation oracle for this polytope. Additionally, once we have found the revenue-optimal reduced form in $F(\mathcal{F}, \mathcal{D})$, we need some way of implementing it. As we know that every corner of $F(\mathcal{F}, \mathcal{D})$ can be implemented by an especially simple allocation rule, we would like a way to decompose a given feasible reduced form into an explicit convex combination of corners (which then corresponds to a distribution over simple virtual VCG allocation rules). In this section, we provide both algorithms. For now, we will not worry about the running time of our algorithms, but just provide a generic framework that applies to all settings. In Section 4.3 we will describe how to approximately implement these algorithms efficiently with high probability obtaining an FPRAS with only black box access to an implementation of the VCG allocation rule. The same algorithms with the obvious modifications also apply to "second-order reduced forms", using the techniques of Section 4.5.

4.2.1 Separation Oracle

Grötschel et al. [GLS81] 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. The characterization result of Section 4.1 essentially tells us how to optimize a linear function in $F(\mathcal{F}, \mathcal{D})$, but not in an inefficient manner. Therefore, our separation oracle can be obtained using the result by Grötschel et al. As our goal for this section is to provide a generic framework, we provide detailed proofs here so that we can conveniently modify them in later sections, in which we provide an efficient construction of the separation oracle.

We know from Corollary 3 that if a reduced form $\vec{\pi}$ is infeasible, then there is

some weight vector $\vec{w} \in [-1,1]^{n\sum_{i=1}^{m}|T_i|}$ such that $\vec{\pi} \cdot \vec{w} > W_{\mathcal{F}}(\vec{w})$. Finding such a weight vector explicitly gives us a hyperplane separating $\vec{\pi}$ from $F(\mathcal{F},\mathcal{D})$, provided we can also compute $W_{\mathcal{F}}(\vec{w})$. So consider the function:

$$g_{\vec{\pi}}(\vec{w}) = W_{\mathcal{F}}(\vec{w}) - \vec{\pi} \cdot \vec{w}.$$

We know that $\vec{\pi}$ is feasible if and only if $g_{\vec{\pi}}(\vec{w}) \geq 0$ for all $\vec{w} \in [-1,1]^{n\sum_{i=1}^{m}|T_i|}$. So the goal of our separation oracle SO is to minimize $g_{\vec{\pi}}(\vec{w})$ over the hypercube, and check if the minimum is negative. If negative, the reduced form is infeasible, and the minimizer bears witness. Otherwise, the reduced form is feasible. To write a linear program to minimize $g_{\vec{\pi}}(\vec{w})$, recall that $W_{\mathcal{F}}(\vec{w}) = \max_{\vec{x}\in F(\mathcal{F},\mathcal{D})}\{\vec{x} \cdot \vec{w}\}$, so $g_{\vec{\pi}}(\vec{w})$ is a piece-wise linear function. Using standard techniques, we could add a variable, t, for $W_{\mathcal{F}}(\vec{w})$, add constraints to guarantee that $t \geq \vec{x} \cdot \vec{w}$ for all $\vec{x} \in F(\mathcal{F},\mathcal{D})$, and minimize $t - \vec{\pi} \cdot \vec{w}$. As this is a burdensome number of constraints, we will use an internal separation oracle $\widehat{SO}$, whose job is simply to verify that $t \geq \vec{x} \cdot \vec{w}$ for all $\vec{x} \in F(\mathcal{F},\mathcal{D})$, and output a violating hyperplane otherwise.

To implement $\widehat{SO}$, let $R_{\mathcal{F}}(\vec{w})$ denote the reduced form of $VVCG_{\mathcal{F}}(\vec{w})$. Then we know that $R_{\mathcal{F}}(\vec{w}) \cdot \vec{w} \geq \vec{x} \cdot \vec{w}$ for all $\vec{x} \in F(\mathcal{F},\mathcal{D})$. So if any equation of the form $\vec{x} \cdot \vec{w} \leq t$ is violated, then certainly $R_{\mathcal{F}}(\vec{w}) \cdot \vec{w} \leq t$ is violated. Therefore, for an input $\vec{w}, t$, we need only check a single constraint of this form. So let $\widehat{SO}(\vec{w}, t)$ output "yes" if $R_{\mathcal{F}}(\vec{w}) \cdot \vec{w} \leq t$, and output the violated hyperplane $R_{\mathcal{F}}(\vec{w}) \cdot \vec{z} - y \leq 0$ otherwise. $\widehat{SO}$ allows us to reformulate a more efficient linear program to minimize $g_{\vec{\pi}}(\vec{w})$.

So our separation oracle SO to check if $\vec{\pi} \in F(\mathcal{F},\mathcal{D})$ is as follows: run the linear program of Figure 4-1 to minimize $g_{\vec{\pi}}(\vec{w})$. Let the optimum output by the LP be $t^*, \vec{w}^*$. If the value of the LP is negative, we know that $\vec{w}^* \cdot \vec{\pi} > t^* = W_{\mathcal{F}}(\vec{w}^*)$, and we have our violated hyperplane. Otherwise, the reduced form is feasible, so we output "yes."

We conclude this section with a lemma relating the bit complexity of the corners of $F(\mathcal{F},\mathcal{D})$ to the bit complexity of the output of our separation oracle. This is handy for efficiently implementing our algorithms in later sections. We make use a standard

Variables:

- t, denoting the value of $W_{\mathcal{F}}(\vec{w})$.
- $w_{ij}(A)$ for all bidders i, items j, and types $A \in T_i$.

Constraints:

- $-1 \leq w_{ij}(A) \leq 1$ for all bidders i, items j, and types $A \in T_i$, guaranteeing that the weights lie in $[-1,1]^{n\sum_{i=1}^{m}|T_i|}$.
- $\widehat{SO}(\vec{w},t) =$ "yes," guaranteeing that $t \geq W_{\mathcal{F}}(\vec{w})$.

Minimizing:

- $t - \vec{\pi}\cdot\vec{w}$ (this is $g_{\vec{\pi}}(\vec{w})$ provided $t = W_{\mathcal{F}}(\vec{w})$).

Figure 4-1: A Linear Program to minimize $g_{\vec{\pi}}(\vec{w})$.

property of the Ellipsoid algorithm (see Theorem 1).

Lemma 5. *If all coordinates of each corner of $F(\mathcal{F},\mathcal{D})$ are rational numbers of bit complexity ℓ, then every coefficient of any hyperplane output by SO is a rational number of bit complexity poly$(n\sum_{i=1}^{m}|T_i|,\ell)$.*

Proof. The dimension of the LP (shown in Figure 4-1) used to run SO is $n\sum_{i=1}^{m}|T_i|$. Every constraint of the linear program that is not part of $\widehat{SO}$ has bit complexity $O(1)$, and the coefficients of every hyperplane output by $\widehat{SO}$ have bit complexity ℓ by our hypothesis. (Recall our discussion in Section 4.1.) So by the theory of Gaussian elimination, the coordinates of all corners of the LP of Figure 4-1 are rational numbers of bit complexity poly$(n\sum_{i=1}^{m}|T_i|,\ell)$. Now recall the third property of the Ellipsoid algorithm from Theorem 1. □

4.2.2 Decomposition Algorithm via a Corner Oracle

We provide an algorithm for writing a feasible reduced-form as a convex combination of corners of $F(\mathcal{F},\mathcal{D})$, i.e. reduced forms of simple virtual VCG allocation rules. A decomposition algorithm for arbitrary polytopes P is already given in Theorem 2, and the only required ingredients for the algorithm is a separation oracle for P, *corner oracle* for P, and bound b on the bit complexity of the coefficients of any hyperplane

that can possibly be output by the separation oracle. The goal of this section is to define both oracles and determine b for our setting. Before stating the result, let us specify the required functionality of the corner oracle.

The corner oracle for polytope P takes as input k (where k is at most the dimension, in our case $n\sum_i |T_i|$) hyperplanes $H_1, \ldots, H_k$ (whose coefficients are all rational numbers of bit complexity b) and has the following behavior: If no hyperplane intersects P in its interior and there is a corner of P that lies in all hyperplanes, then such a corner is output. Otherwise, the behavior may be arbitrary. So all we need to do is define CO and SO, and provide a bound on the bit complexity of the hyperplanes output by SO. We've already defined SO and bounded the bit complexity of hyperplanes output by it by $\text{poly}(n\sum_{i=1}^{m} |T_i|, \ell)$, where ℓ is the maximum number of bits needed to represent a coordinate in a corner of $F(\mathcal{F}, \mathcal{D})$ (see Lemma 5 of Section 4.2.1). So now we define CO and state its correctness in Theorem 11 whose proof is in Appendix A.1. In the last line, CO outputs the weights $\vec{w}'$ as well so that we can actually implement the reduced form that is output.

Algorithm 2 Corner Oracle for $F(\mathcal{F}, \mathcal{D})$

Input: Hyperplanes $(\vec{w}_1, h_1), \ldots, (\vec{w}_a, h_a)$, $a \leq n\sum_{i=1}^{m} |T_i|$.
Set $\vec{w} = \sum_{j=1}^{a} \frac{1}{a}\vec{w}_j$.
Use the tie-breaking rule of Section 4.1 (stated formally in Lemma 4) on $\vec{w}$ to obtain $\vec{w}'$.
Output the reduced form of $VVCG_{\mathcal{F}}(\vec{w}')$, as well as $\vec{w}'$.

Theorem 11. *The Corner Oracle of Algorithm 4.2.1 correctly outputs a corner of $F(\mathcal{F}, \mathcal{D})$ contained in $\cap_{j=1}^{a} H_j$ whenever the hyperplanes $H_1, \ldots, H_a$ are boundary hyperplanes of $F(\mathcal{F}, \mathcal{D})$ and $\cap_{j=1}^{a} H_j$ contains a corner. Furthermore, if all coordinates of all H_j are rational numbers of bit complexity b, and $\Pr[t_i = A]$ is a rational number of bit complexity ℓ for all $i, A \in T_i$, then every coordinate of the weight vector $\vec{w}'$ is a rational number of bit complexity poly$(n\sum_{i=1}^{m} |T_i|, b, \ell)$.*

4.3 Efficient Implementation of Algorithms for Reduced Forms

In this section, we show how to approximately implement the separation oracle (SO) of Section 4.2.1 and the corner oracle (CO) of Section 4.2.2 efficiently with high probability, thereby obtaining also an approximate decomposition algorithm for $F(\mathcal{F}, \mathcal{D})$. We begin by bounding the runtime of an exact implementation, showing that it is especially good when $\mathcal{D}$ is a uniform (possibly non-product) distribution of small support. As above, $A_{\mathcal{F}}$ denotes an algorithm that implements the VCG allocation rule with respect to feasibility constraints $\mathcal{F}$, and $rt_{\mathcal{F}}(b)$ denotes the runtime of $A_{\mathcal{F}}$ when each input weight has bit complexity b.

4.3.1 Exact Implementation

The only tricky step in implementing SO and CO is computing $R_{\mathcal{F}}(\vec{w})$ for a given $\vec{w}$. A simple approach is to just enumerate every profile in the support of $\mathcal{D}$ and check if $VVCG_{\mathcal{F}}(\vec{w})$ awards bidder i item j. This can be done in time polynomial in the cardinality $|\mathcal{D}|$ of the support of $\mathcal{D}$, the bit complexity ℓ of the probabilities used by $\mathcal{D}$ and $rt_{\mathcal{F}}(\text{poly}(b, \ell))$, where b is the bit complexity of $\vec{w}$'s coordinates. So, if b is an upper bound on the bit complexity of the coordinates of the weight vectors $\vec{w}$ for which $R_{\mathcal{F}}(\vec{w})$ is computed in an execution of SO (CO), then SO (CO) can be implemented in time polynomial in $n \sum_i |T_i|, |\mathcal{D}|$, ℓ, b, c, and $rt_{\mathcal{F}}(\text{poly}(b, \ell))$, where c is the bit complexity of the numbers in the input of SO (CO). Alone, this result is not very helpful as we can do much more interesting computations in time polynomial in $|\mathcal{D}|$, including exactly solve MDMDP [DW12]. The interesting corollary is that when $\mathcal{D}$ is a (possibly correlated) uniform distribution over a collection of profiles (possibly with repetition) whose number is polynomial in $n \sum_i |T_i|$, the runtime of all algorithms of Section 4.2 becomes polynomial in $n \sum_i |T_i|$, c, and $rt_{\mathcal{F}}(\text{poly}(n \sum_i |T_i|, c))$, where c is the bit complexity of the numbers in the input to these algorithms. Corollaries 15 and 16 in Appendix A.2 quantify this statement precisely. It is these corollaries that

enable an efficient approximation for arbitrary distributions in the next section.

4.3.2 Approximate Implementation

Now, we show how to "approximately implement" both algorithms in time polynomial in only $\sum_{i=1}^{m} |\mathcal{D}_i|$, where $|\mathcal{D}_i|$ is the cardinality of the support of $\mathcal{D}_i$, using the results of Section 4.3.1. But we need to use the right notion of approximation. Simply implementing both algorithms approximately, e.g. separating out reduced forms that are not even approximately feasible and decomposing reduced forms that are approximately feasible, might not get us very far, as we could lose the necessary linear algebra to solve LPs. So we use a different notion of approximation. We compute a polytope P' that, with high probability, is a "good approximation" to $F(\mathcal{F}, \mathcal{D})$ in the sense that instead of optimizing over $F(\mathcal{F}, \mathcal{D})$ we can optimize over P' instead. Then we implement both the separation and the decomposition algorithms for P' exactly so that their running time is polynomial in n, $\sum_{i=1}^{m} |T_i|$, c and $rt_{\mathcal{F}}(\text{poly}(n \sum_{i=1}^{m} |T_i|, c))$, where c is the number of bits needed to describe a coordinate of the input to these algorithms.

Approach: So how can we compute an approximating polytope? Our starting point is a natural idea: Given an arbitrary distribution $\mathcal{D}$, we can sample profiles $P_1, \ldots, P_k$ from $\mathcal{D}$ independently at random and define a new distribution $\mathcal{D}'$ that samples a profile uniformly at random from $P_1, \ldots, P_k$ (i.e. chooses each P_i with probability $1/k$). Clearly as $k \to \infty$ the polytope $F(\mathcal{F}, \mathcal{D}')$ should approximate $F(\mathcal{F}, \mathcal{D})$ better and better. The question is how large k should be taken for a good approximation. If taking k to be polynomial in $n \sum_{i=1}^{m} |T_i|$ suffices, then Section 4.3.1 (specfically, Corollaries 15 and 16 of Appendix A.2) also implies that we can implement both the separation and the decomposition algorithms for $F(\mathcal{F}, \mathcal{D}')$ in the desired running time.

However this approach fails, as some types may very well have $\Pr[t_i = A] << \frac{1}{\text{poly}(n \sum_{i=1}^{m} |T_i|)}$. Such types likely wouldn't even appear in the support of $\mathcal{D}'$ if k scales polynomially in $n \sum_{i=1}^{m} |T_i|$. So how would then the proxy polytope $F(\mathcal{F}, \mathcal{D}')$ inform

us about $F(\mathcal{F}, \mathcal{D})$ in the corresponding dimensions? To cope with this, for each bidder i and type $A \in T_i$, we take an additional k' samples from $\mathcal{D}_{-i}$ and set $t_i = A$. $\mathcal{D}'$ still picks uniformly at random from all $k + k' \sum_{i=1}^{m} |T_i|$ profiles.

Now here is what we can guarantee. In Corollary 6 (stated and proved in Section 4.4.1), we show that with high probability every $\vec{\pi}$ in $F(\mathcal{F}, \mathcal{D})$ has some $\vec{\pi}' \in F(\mathcal{F}, \mathcal{D}')$ with $|\vec{\pi} - \vec{\pi}'|_\infty$ small. This is done by taking careful concentration and union bounds. In Corollary 7 (stated and proved in Section 4.4.1), we show the converse: that with high probability every $\vec{\pi}' \in F(\mathcal{F}, \mathcal{D}')$ has some $\vec{\pi} \in F(\mathcal{F}, \mathcal{D})$ with $|\vec{\pi} - \vec{\pi}'|_\infty$ small. This requires a little more care as the elements of $F(\mathcal{F}, \mathcal{D}')$ are not fixed a priori (i.e. before taking samples from $\mathcal{D}$ to define $\mathcal{D}'$), but depend on the choice of $\mathcal{D}'$, which is precisely the object with respect to which we want to use the probabilistic method. We resolve this apparent circularity by appealing to some properties of the algorithms of Section 4.2 (namely, bounds on the bit complexity of any output of SO and CO). Finally, in Theorems 12 and 13 (stated below and proved in Section 4.4.2), we put our results together to prove that our approximations behave as desired while taking k and k' both polynomial in $n \sum_{i=1}^{m} |T_i|$, thereby achieving the desired runtime.

In the following theorems, Algorithm 3 refers to a pre-processing algorithm (see Section 4.4.2) that explicitly chooses k and k', both polynomial in $n \sum_i |T_i|$, so that the polytopes $F(\mathcal{F}, \mathcal{D})$ and $F(\mathcal{F}, \mathcal{D}')$ are close with high probability. Algorithm 4 refers to a decomposition algorithm (Section 4.4.2) combining the geometric algorithm of in Section 2.8 with some bookkeeping to decompose any reduced form in $F(\mathcal{F}, \mathcal{D}')$ into an explicit distribution over simple virtual VCG allocation rules.

Theorem 12. *Given the choice of k, k' and $\mathcal{D}'$ in Algorithm 3, the following are true with probability at least $1 - e^{-\Omega(n \sum_{i=1}^{m} |T_i|/\epsilon)}$:*

1. *For all $\vec{\pi} \in F(\mathcal{F}, \mathcal{D})$, there is a $\vec{\pi}' \in F(\mathcal{F}, \mathcal{D}')$ with $|\vec{\pi} - \vec{\pi}'|_\infty \leq \epsilon$.*

2. *For all $\vec{\pi}' \in F(\mathcal{F}, \mathcal{D}')$, there is a $\vec{\pi} \in F(\mathcal{F}, \mathcal{D})$ with $|\vec{\pi} - \vec{\pi}'|_\infty \leq \epsilon$.*

Moreover the separation oracle of Section 4.2.1 for feasibility set $\mathcal{F}$ and distribution $\mathcal{D}'$ runs in time polynomial in n, $\sum_{i=1}^{m} |T_i|$, $1/\epsilon$, c, and $rt_{\mathcal{F}}(poly(n \sum_{i=1}^{m} |T_i|, \log 1/\epsilon, c))$, where c is the bit complexity of the coordinates of its input.

Theorem 13. *Algorithm 4 has the following property on input $\vec{\pi}' \in F(\mathcal{F}, \mathcal{D}')$ with probability at least $1 - e^{-O(n \sum_{i=1}^{m} |T_i|/\epsilon)}$: Let $\vec{\pi}$ denote the reduced form of the output allocation rule when consumers are sampled from $\mathcal{D}$. Then $|\vec{\pi} - \vec{\pi}'|_\infty \leq \epsilon$. Furthermore, the running time of the algorithm is polynomial in $n, \sum_{i=1}^{m} |T_i|, 1/\epsilon$, c, and $rt_{\mathcal{F}}(poly(n \sum_{i=1}^{m} |T_i|, \log 1/\epsilon, c))$, where c is the bit complexity of the coordinates of its input.*

Remark 1. *Observe that, despite the non-canonical way Theorems 12 and 13 are stated for FPRASs, the dependence of the running time on the approximation error and the failure probability is the typical one. Namely, using the stated results as black box we can simultaneously achieve error probability at most η and approximation error at most ϵ in time polynomial in $\log 1/\eta$, $1/\epsilon$, n, $\sum_i |T_i|$, c and $rt_{\mathcal{F}}(poly(n \sum_{i=1}^{m} |T_i|, \log 1/\epsilon, \log\log(1/\eta), c))$, where c is the bit complexity of the coordinates of the input to our algorithms.*

4.4 Details for Approximate Implementation

Notation We will use the following notation throughout this section: k is the number of samples taken directly from $\mathcal{D}$, k' is the number of samples taken from $\mathcal{D}_{-i}$ after fixing $t_i = A$ for all $i, A \in T_i$, k'' is the total number of samples taken (i.e. $k'' = k + k' \sum_i |T_i|$) and $\mathcal{D}'$ is the distribution that samples one of the k'' sampled profiles uniformly at random. We also make use of the following, standard Chernoff bound:

Theorem 14. *(Hoeffding [Hoe63]) Let $X_1, \ldots, X_n$ be independent random variables in $[0, 1]$, and let $X = \sum_i X_i/n$. Then $\Pr[|X - \mathbb{E}[X]| > t] \leq 2e^{-2t^2 n}$.*

4.4.1 An Approximate Polytope

Every point in $F(\mathcal{F}, \mathcal{D})$ is close to some point in $F(\mathcal{F}, \mathcal{D}')$

The desired claim is stated at the end of the section as Corollary 6, and the proof is obtained by a series of small technical lemmas. Throughout this section, we will be interested in whether a profile with $t_i = A$ in the support of $\mathcal{D}'$ is *genuine* from the perspective of bidder i (i.e. it was sampled from $\mathcal{D}$ without conditioning on anything except perhaps $t_i = A$) or *biased* (i.e. it was sampled by conditioning on $t_{i'} = A'$ for some $i' \neq i$). Now let us fix an allocation rule M with reduced form $\vec{\pi}$ if bidders are sampled from $\mathcal{D}$. What does the reduced form of M look like for bidders sampled from $\mathcal{D}'$? The expected probability (over the randomness in the types of the other bidders) that bidder i receives item j conditioning on $t_i = A$ on a genuine from bidder i's perspective profile is exactly $\pi_{ij}(A)$. However, if the profile is biased, the probability that bidder i receives item j might have nothing to do with $\pi_{ij}(A)$. So for a fixed $\mathcal{D}'$, we'll let $G_i(A)$ denote the (random) set of profiles in the support of $\mathcal{D}'$ with $t_i = A$ that were obtained genuinely from the perspective of bidder i, and $B_i(A)$ denote the set of profiles with $t_i = A$ that are biased from the perspective of bidder i.

Lemma 6. *Fix i and $A \in T_i$, let M be any allocation rule, and let $\mathcal{D}'$ be such that*

$|G_i(A)| = x$ and $|B_i(A)| = z$ (i.e. condition on $|G_i(A)| = x, |B_i(A)| = z$ and then sample $\mathcal{D}'$). Then over the randomness in generating $\mathcal{D}'$, for all items j and all $t \leq 1$, if $\vec{\pi}'$ denotes the reduced form of M when bidders are sampled from $\mathcal{D}'$ and $\vec{\pi}$ denotes the reduced form of M when bidders are sampled from $\mathcal{D}$, we have:

$$\Pr\left[|\pi_{ij}(A) - \pi'_{ij}(A)| > t + \frac{z}{x}\right] \leq 2e^{-2t^2 x}.$$

Proof. Label the profiles in $G_i(A)$ as $P_1, \ldots, P_x$ and $B_i(A)$ as $P_{x+1}, \ldots, P_{x+z}$, and let X_a be the random variable denoting the probability that M awards item j to bidder i on profile P_a. Then for all $1 \leq a \leq x$ we have $\mathbb{E}[X_a] = \pi_{ij}(A)$. For all $a > x$, we have $0 \leq \mathbb{E}[X_a] \leq 1$. As $\pi'_{ij}(A) = \frac{1}{z+x}\sum_a X_a$, we see that:

$$\pi_{ij}(A) - \frac{z}{x+z} \leq \frac{x}{x+z}\pi_{ij}(A) \leq \frac{x}{x+z}\pi_{ij}(A) + \frac{1}{x+z}\sum_{i=1}^{z}\mathbb{E}[X_{x+i}] = \mathbb{E}[\pi'_{ij}(A)] \leq \frac{x}{x+z}\pi_{ij}(A) + \frac{z}{x+z} \leq \pi_{ij}(A) + \frac{z}{x+z}.$$

So $|\mathbb{E}[\pi'_{ij}(A)] - \pi_{ij}(A)| \leq \frac{z}{x+z}$. Therefore, the triangle inequality tells us that in order to have $|\pi_{ij}(A) - \pi'_{ij}(A)| > t + \frac{z}{x} > t + \frac{z}{z+x}$, we must have $|\pi'_{ij}(A) - \mathbb{E}[\pi'_{ij}(A)]| > t$. As $\pi'_{ij}(A)$ is the average of $x + z$ independent trials, by the Hoeffding inequality, this happens with probability at most $2e^{-2t^2 x}$. □

Lemma 7. *For any $i, A \in T_i$, $|G_i(A)| \geq k'$. Furthermore, if $k > k' \sum_{i' \neq i} |T_{i'}|$, for all $x \leq 1$ we have:*

$$\Pr\left[|B_i(A)| > \left(2x + \frac{k'}{k}\right)\sum_{i' \neq i}|T_{i'}| \cdot |G_i(A)|\right] \leq 4e^{-2x^2 k' n \sum_{i' \neq i}|T_{i'}|}.$$

Proof of Lemma 7: The first claim is obvious, as we fix $t_i = A$ in exactly k' profiles. For the second claim, there are $k'(\sum_{i' \neq i}|T_{i'}|)$ independent chances to get a profile in

$B_i(A)$. Each chance occurs with probability $q = \Pr[t_i = A]$. There are k independent chances to get additional profiles in $G_i(A)$, and each occurs with probability q. Therefore, we get that $\mathbb{E}[|B_i(A)|] = qk' \sum_{i' \neq i} |T_i|$ and $\mathbb{E}[|G_i(A)|] = k' + qk$. Applying the Hoeffding inequality, we get

$$\Pr\left[\left| \, |B_i(A)| - qk' \sum_{i' \neq i} |T_{i'}| \, \right| > xk' \sum_{i' \neq i} |T_{i'}| \right] \leq 2e^{-x^2 k' \sum_{i' \neq i} |T_{i'}|},$$

and

$$\Pr\left[\Big| |G_i(A)| - (k' + qk) \Big| > xk \right] \leq 2e^{-x^2 k}.$$

Then since $k \geq k' \sum_{i' \neq i} |T_{i'}|$, by union bound, we get that for any $x \leq 1$, with probability at least $1 - 4e^{-2x^2 k' \sum_{i' \neq i} |T_{i'}|}$ we have the following two inequalities:

$$|B_i(A)| \leq qk' \sum_{i' \neq i} |T_{i'}| + xk' \sum_{i' \neq i} |T_{i'}|$$

$$|G_i(A)| \geq k' + \max\{0, (q - x)k\} \tag{4.1}$$

which imply the following two inequalities by ignoring one of the positive terms on the right-hand side of Equation (4.1):

$$|B_i(A)| \leq (q + x) \sum_{i' \neq i} |T_{i'}| \cdot |G_i(A)| \tag{4.2}$$

$$|B_i(A)| \leq \frac{q + x}{q - x} \cdot \frac{k' \sum_{i' \neq i} |T_{i'}|}{k} |G_i(A)| \quad \text{(we only use this when } q > x) \tag{4.3}$$

When $q \leq x + \frac{k'}{k}$, Equation (4.2) gives a better bound. Otherwise, Equation (4.3) gives a better bound. As q decreases, the bound from Equation (4.2) only gets better. Likewise, as q increases, the bound from Equation (4.3) only gets better. So for any q, one of the bounds will yield:

$$|B_i(A)| \leq \left(2x + \frac{k'}{k} \right) \sum_{i' \neq i} |T_{i'}| \cdot |G_i(A)|$$

as desired.□

Corollary 4. *Let M be any allocation rule and assume $k > k' \sum_{i' \neq i} |T_{i'}|$. Then for all items j, bidders i, types $A \in T_i$, and all $t \leq 1$, if $\vec{\pi}'$ denotes the reduced form of M when bidders are sampled from $\mathcal{D}'$ and $\vec{\pi}$ denotes the reduced form of M when bidders are sampled from $\mathcal{D}$, we have:*

$$\Pr\left[|\pi_{ij}(A) - \pi'_{ij}(A)| > t + \left(2t + \frac{k'}{k}\right) \sum_{i' \neq i} |T_{i'}|\right] \leq 6e^{-2t^2 k'}.$$

Proof of Corollary 4: Lemma 7 says that with probability at least $1 - 4e^{-2x^2 k' \sum_{i' \neq i} |T_{i'}|} \geq 1 - 4e^{-2x^2 k'}$, $\mathcal{D}'$ is such that $|B_i(A)| \leq (2x + \frac{k'}{k}) \sum_{i' \neq i} |T_{i'}| \cdot |G_i(A)|$ and $|G_i(A)| \geq k'$. For such $\mathcal{D}'$, the bound given by Lemma 6 is:

$$\Pr\left[|\pi_{ij}(A) - \pi'_{ij}(A)| > t + \left(2x + \frac{k'}{k}\right) \sum_{i' \neq i} |T_{i'}|\right] \leq 2e^{-2t^2 k'}.$$

So after taking a union bound and setting $x = t$ we get the desired claim. □

Corollary 5. *Let M be any allocation rule and assume $k > k' \sum_{i' \neq i} |T_{i'}|$. Then if $\vec{\pi}'$ denotes the reduced form of M when bidders are sampled from $\mathcal{D}'$ and $\vec{\pi}$ denotes the reduced form when bidders are sampled from $\mathcal{D}$, we have:*

$$\Pr\left[|\vec{\pi} - \vec{\pi}'|_\infty > t + \left(2t + \frac{k'}{k}\right) \sum_{i} |T_i|\right] \leq 6n \sum_{i=1}^{m} |T_i| e^{-2t^2 k'}.$$

Proof of Corollary 5: Use Corollary 4, observe that $\sum_{i'} |T_{i'}| > \sum_{i' \neq i} |T_{i'}|$ for all i, and take a union bound over all $j, i, A \in T_i$. □

Corollary 6. *Assume $k > k' \sum_i |T_i|$. Then for all $t \leq 1$, with probability at least*

$$1 - 6n \sum_{i=1}^{m} |T_i| e^{-2t^2 k' - n \sum_{i=1}^{m} |T_i| \ln t},$$

for every $\vec{\pi} \in F(\mathcal{F}, \mathcal{D})$, there is a $\vec{\pi}' \in F(\mathcal{F}, \mathcal{D}')$ with $|\vec{\pi} - \vec{\pi}'|_\infty \leq 2t + (2t + \frac{k'}{k}) \sum_{i=1}^{m} |T_i|$.

Proof of Corollary 6: Consider an t-ℓ_∞ cover of $F(\mathcal{F}, \mathcal{D})$ such that every point in $F(\mathcal{F}, \mathcal{D})$ is within ℓ_∞ distance t of a point in the cover. There is certainly a cover

that uses at most $\left(\frac{1}{t}\right)^{n\sum_{i=1}^{m}|T_i|}$ points, as there is a cover of the entire hypercube using this many points. If for every point $\vec{x}$ in the cover, there is a point $\vec{z} \in F(\mathcal{F}, \mathcal{D}')$, such that $|\vec{x} - \vec{z}|_\infty \leq t + (2t + \frac{k'}{k})\sum_i |T_i|$, then clearly for every point $\vec{\pi}$ in $F(\mathcal{F}, \mathcal{D})$, there is a point $\vec{\pi}' \in F(\mathcal{F}, \mathcal{D}')$ such that $|\vec{\pi} - \vec{\pi}'|_\infty \leq 2t + (2t + \frac{k'}{k})\sum_i |T_i|$ by the triangle inequality. So we simply take a union bound over all $e^{-n\sum_{i=1}^{m}|T_i|\ln t}$ points in the cover and apply Corollary 5 to conclude the proof. □

Every point in $F(\mathcal{F}, \mathcal{D}')$ is close to some point in $F(\mathcal{F}, \mathcal{D})$

In the previous section, we showed that for all $\vec{\pi} \in F(\mathcal{F}, \mathcal{D})$ there is a nearby $\vec{\pi}' \in F(\mathcal{F}, \mathcal{D}')$ using the probabilistic method over the choice of $\mathcal{D}'$. In this section, we want to show the other direction, namely that for any reduced form $\vec{\pi}' \in F(\mathcal{F}, \mathcal{D}')$ there is a nearby reduced form $\vec{\pi} \in F(\mathcal{F}, \mathcal{D})$. However, it is not clear how to use the probabilistic method over the choice of $\mathcal{D}'$ to prove this, as for $\vec{\pi}' \in F(\mathcal{F}, \mathcal{D}')$ the allocation rule that implements $\vec{\pi}'$ is heavily dependent on $\mathcal{D}'$, which is the object with respect to which we plan to apply the probabilistic method. To go around this circularity we show that after fixing k and k', but before selecting $\mathcal{D}'$, there are not too many allocation rules that could possibly implement a reduced form that is a corner of $F(\mathcal{F}, \mathcal{D}')$. Specifically, we show that the reduced form with respect to $\mathcal{D}'$ of any simple virtual VCG allocation rule is equivalent to one whose functions only output rational numbers with bit complexity that only depends on k, k' and the dimension. In particular, regardless of $\mathcal{D}'$, there is an a-priori fixed set of allocation rules that implement the corners of $F(\mathcal{F}, \mathcal{D}')$ whose cardinality depends only on k, k' and the dimension. So we can still use concentration of measure to argue that the reduced form π' of every corner of $F(\mathcal{F}, \mathcal{D}')$ has a nearby reduced form $\pi \in F(\mathcal{F}, \mathcal{D})$. And, as every point in $F(\mathcal{F}, \mathcal{D}')$ is a convex combination of the corners of $F(\mathcal{F}, \mathcal{D}')$, this suffices to prove the desired claim. Our starting point is the following observation.

Lemma 8. *Let $\vec{\pi}$ be the reduced form of a simple virtual VCG allocation with respect to $\mathcal{D}'$. Then each $\pi_{ij}(A)$ is a rational number of bit complexity $O(\log k'')$.*

Proof of Lemma 8: In every simple virtual VCG allocation, the probability that

bidder i gets item j on profile P is always 1 or 0. Therefore, if $t_i = A$ in exactly x profiles, and bidder i receives item j in exactly x' of those profiles, $\pi_{ij}(A) = x'/x$. As $x' \leq x \leq k''$, this value clearly has bit complexity $O(\log k'')$. $\square$

Given Lemma 8 the corners of $F(\mathcal{F}, \mathcal{D}')$ belong to a set of at most $(k'')^{O(n\sum_i |T_i|)}$ reduced forms that is independent of $\mathcal{D}'$. Still the allocation rules that implement those reduced forms may vary depending on $\mathcal{D}'$. We show that this can be mitigated by appealing to the correctness of the decomposition algorithm of Section 4.2.2.

Lemma 9. *Suppose that the allocation rule M implements a corner $\vec{\pi}$ of $F(\mathcal{F}, \mathcal{D}')$. Then there is a virtual VCG allocation rule $VVCG(\{f_i\}_{i\in[m]})$ whose reduced form with respect to $\mathcal{D}'$ is exactly $\vec{\pi}$ and such that each f_i only outputs rational numbers of bit complexity $f_c(n\sum_{i=1}^m |T_i| \log k'')$, where $f_c(\cdot)$ is a polynomial function.*

Moreover, for any input $\vec{\pi} \in F(\mathcal{F}, \mathcal{D}')$ to the decomposition algorithm of Section 4.2.2 for $\mathcal{D}'$, the output decomposition uses simple virtual VCG allocation rules whose weight functions only output rational numbers of complexity $f_c(n\sum_{i=1}^m |T_i| \log k'')$.

Proof of Lemma 9: For the first part of the theorem, suppose that a corner $\vec{\pi}$ of $F(\mathcal{F}, \mathcal{D}')$ is fed as input to the decomposition algorithm of Section 4.2.2. From the correctness of this algorithm it follows that the output decomposition consists of a single reduced form, namely $\vec{\pi}$ itselft, as $\vec{\pi}$ is a corner of $F(\mathcal{F}, \mathcal{D}')$. The decomposition algorithm in Section 2.8 (which is at the heart of the decomposition algorithm of Section 4.2.2) has the property that every corner used in the output decomposition will always be an output of the corner oracle. So let us try to argue that all reduced forms that are possibly output by the corner oracle can be implemented by a small set of allocation rules that does not depend on $\mathcal{D}'$. We use the following lemma:

Lemma 10. *On any input $\vec{\pi}$, the coefficients of the hyperplane output by the separation oracle SO of Section 4.2.1 using $\mathcal{D}'$ as the bidder-type distribution are rational numbers of bit complexity $f_s(n\sum_{i=1}^m |T_i| \log k'')$, where $f_s(\cdot)$ is a polynomial function.*

Proof. By Lemma 8, we know that all coordinates of all corners of $F(\mathcal{F}, \mathcal{D}')$ can be described using at most $O(\log k'')$ bits. Lemma 5 now tells us that SO will only output rational numbers of bit complexity $\text{poly}(n\sum_{i=1}^m |T_i|, \log k'')$. $\square$

Now let's go back to the corner oracle. By Lemma 10 the weights input to the corner oracle will always be rational numbers of bit complexity $f_s(n\sum_{i=1}^m |T_i| \log k'') = \text{poly}(n\sum_{i=1}^m |T_i|, \log k'')$. As the number of hyperplanes input to the corner oracle will never be more than $n\sum_{i=1}^m |T_i|$, the weights obtained by averaging in step 2 of the corner oracle require at most an additional $O(\log(n\sum_{i=1}^m |T_i|))$ bits. Given the above and that, for all i, A, $\Pr[t_i = A]$ is a multiple of $1/k''$ and hence has bit complexity $O(\log k'')$, the tie-breaking rule in step 3 of the corner oracle results in a weight vector whose coordinates have bit complexity $\text{poly}(n\sum_{i=1}^m |T_i|, \log k'')$. As $\Pr[t_i = A]$ is a multiple of $1/k''$ for all i, A, transforming from the weight vector representation of the simple VCG mechanism computed by the corner oracle to the weight function representation adds at most an additional $O(\log k'')$ bits per weight.

The second part of the lemma is already implied by the above discussion. As we noted above the decomposition algorithm in Section 2.8 (which is at the heart of the decomposition algorithm of Section 4.2.2) has the property that every corner used in the output decomposition will always be an output of the corner oracle. And we argued that the corner oracle for $\mathcal{D}'$ outputs simple virtual VCG allocation rules whose weight functions only output rationals of bit complexity as bounded above. □

Lemma 9 implies that, before we have sampled $\mathcal{D}'$ but after we have chosen k and k', there is a fixed set $\mathcal{K}$ of at most $4^{n\sum_{i=1}^m |T_i| f_c(n\sum_{i=1}^m |T_i| \log k'')}$ simple virtual VCG allocation rules (namely those whose weight functions only output rational numbers of bit complexity $f_c(n\sum_{i=1}^m |T_i| \log k'')$) such that, no matter what $\mathcal{D}'$ is sampled, all corners of $F(\mathcal{F}, \mathcal{D}')$ can be implemented by a simple virtual VCG allocation rule in $\mathcal{K}$. Moreover, the decomposition algorithm of Section 4.2.2 only uses simple virtual VCG mechanisms from $\mathcal{K}$ in its support. This implies the following.

Corollary 7. *Assume $k > k' \sum_i |T_i|$ and $t \leq 1$. Then, with probability at least*

$$1 - 6n \sum_{i=1}^{m} |T_i| e^{-2t^2 k' + n\sum_{i=1}^m |T_i| f_c(n\sum_{i=1}^m |T_i| \log k'') \ln 4},$$

the following hold, where $f_c(\cdot)$ is a polynomial function:

1. *every* $\vec{\pi}' \in F(\mathcal{F}, \mathcal{D}')$ *has some* $\vec{\pi} \in F(\mathcal{F}, \mathcal{D})$ *with* $|\vec{\pi} - \vec{\pi}'|_\infty \leq t + (2t + \frac{k'}{k}) \sum_i |T_i|$;

2. *if* $\vec{\pi}$ *is the reduced form with respect to* $\mathcal{D}$ *of the distribution over simple virtual VCG allocation rules that is output on input* $\vec{\pi}' \in F(\mathcal{F}, \mathcal{D}')$ *by the decomposition algorithm of Section 4.2.2 for* $\mathcal{D}'$ *then* $|\vec{\pi} - \vec{\pi}'|_\infty \leq t + (2t + \frac{k'}{k}) \sum_i |T_i|$.

Proof of Corollary 7: For a fixed simple virtual VCG allocation rule $M \in \mathcal{K}$, Corollary 5 guarantees that the reduced form of M when consumers are sampled from $\mathcal{D}$, $\vec{\pi}(M)$, and when consumers are sampled from $\mathcal{D}'$, $\vec{\pi}'(M)$, satisfy: $|\vec{\pi}(M) - \vec{\pi}'(M)|_\infty \leq t + (2t + \frac{k'}{k}) \sum_{i=1}^m |T_i|$ with probability at least $1 - 6n \sum_{i=1}^m |T_i| e^{-2t^2 k'}$. In addition, Lemma 9 guarantees that $|\mathcal{K}| \leq 4^{f_c(n \sum_{i=1}^m |T_i| \log k'') n \sum_{i=1}^m |T_i|}$. Because this set is fixed a priori and independent of $\mathcal{D}'$, we may take a union bound over the elements of the set to get that the same claim holds for *all* simple virtual VCG allocation rules in $\mathcal{K}$ with probability at least

$$1 - 6n \sum_{i=1}^m |T_i| e^{-2t^2 k'} 4^{n \sum_{i=1}^m |T_i| f_c(n \sum_{i=1}^m |T_i| \log k'')}.$$

We proceed to show (i) and (ii) conditioning on the above. For (i) we use the first part of Lemma 9 to get that for all $\vec{\pi}' \in F(\mathcal{F}, \mathcal{D}')$, we can write $\vec{\pi}' = \sum_a p_a \vec{\pi}'(M_a)$, where for all a: $M_a \in \mathcal{K}$, $p_a > 0$, and $\sum_a p_a = 1$. If we consider the exact same distribution over simple virtual VCG allocation rules when consumers are sampled from $\mathcal{D}$, the reduced form will be $\vec{\pi} = \sum_a p_a \vec{\pi}(M_a)$. Given that for all $M_a \in \mathcal{K}$ we have $|\vec{\pi}(M_a) - \vec{\pi}'(M_a)|_\infty \leq t + (2t + \frac{k'}{k}) \sum_{i=1}^m |T_i|$, we have $|\vec{\pi} - \vec{\pi}'|_\infty \leq t + (2t + \frac{k'}{k}) \sum_{i=1}^m |T_i|$ as well.

For (ii) the proof is virtually identical. By the second part of Lemma 9 the simple virtual VCG allocation rules in the support of the decomposition belong to the set $\mathcal{K}$. We proceed as above. □

4.4.2 Putting Everything Together

Setting k and k', and sampling $\mathcal{D}'$

Algorithm 3 is a preprocessing algorithm used to set the parameters k and k' that were left free in Sections 4.4.1 and 4.4.1.

Algorithm 3 Pre-processing to generate a proxy distribution $\mathcal{D}'$ for $\mathcal{D}$. The desired ℓ_∞ accuracy is ϵ.

1: Input: $\mathcal{D}$. Denote by $T = \sum_{i=1}^{m} |T_i|$.
2: Set $t = \frac{\epsilon}{6T}, k' = \frac{n^2T^2 f_c(nT)}{t^3}, k = \frac{4k'T}{\epsilon}$
3: Build $\mathcal{D}'$ by sampling k profiles independently from $\mathcal{D}$. For each $i, A \in T_i$, fix $t_i = A$ and sample k' profiles independently from $\mathcal{D}_{-i}$. $\mathcal{D}'$ picks one of the $k + k' \sum_{i=1}^{m} |T_i|$ sampled profiles uniformly at random.
4: Output $t, k, k', \mathcal{D}'$.

Separation Oracle for Approximating Polytope $F(\mathcal{F}, \mathcal{D}')$

We provide the proof Theorem 12.

Proof of Theorem 12: We use the shorthand $T = \sum_{i=1}^{m} |T_i|$. After plugging in our choice of t, k', k, we see that $2t + (2t + \frac{k'}{k}) \sum_{i=1}^{m} |T_i| \leq \epsilon$. So we just have to verify that the probability bounds given by Corollaries 6 and 7 are as desired.

Before plugging in the choice of t, k', k to Corollary 6, we get that the first claim is true with probability at least $1 - 6nTe^{-2t^2k' - nT\ln t}$. As $k' \geq nT/t^3$, this is at least $1 - 6nTe^{-(2/t - \ln(1/t))nT}$. As $1/t$ asymptotically dominates $\ln(1/t)$ as $t \to 0$, this probability becomes $1 - e^{-\Omega(nT/\epsilon)}$ after plugging in our choice of t.

Before plugging in the choice of t, k', k to Corollary 7, we get that the second claim is true with probability at least $1 - 6nTe^{-2t^2k' + nTf_c(nT\log(k+k'))\ln 4}$. Plugging in the choice of k' and k (and observing that $\log(k + k')$ is $O(\log(nT/t))$ this becomes:

$$1 - 6nTe^{-2n^2T^2 f_c(nT)/t \; + \; nTf_c(nT \cdot O(\log(nT/t)))\ln 4}.$$

Therefore, the ratio of the absolute value of the negative term in the exponent to the value of the positive term is $\frac{nT/t}{\text{poly}\log(nT/t)}$, so the negative term dominates asymp-

totically as $t \to 0$. Therefore, the entire probability is $1 - e^{-\Omega(nT/\epsilon)}$ after plugging in our choice of t.

The bound on the running time follows directly from Corollary 15 and our choice of parameters. $\square$

Decomposition Algorithm for Approximating Polytope $F(\mathcal{F}, \mathcal{D}')$

Algorithm 4 describes our decomposition algorithm for $F(\mathcal{F}, \mathcal{D}')$. After stating it, we analyze it.

Algorithm 4 Algorithm for decomposing a reduced form $\vec{\pi}' \in F(\mathcal{F}, \mathcal{D}')$ into a distribution over simple virtual VCG allocations.

1: Input: $\mathcal{F}$, $\mathcal{D}'$, $\vec{\pi}' \in F(\mathcal{F}, \mathcal{D}')$.

2: Run the geometric algorithm in Section 2.8 on $F(\mathcal{F}, \mathcal{D}')$ using the separation oracle of Section 4.2.1 and the corner oracle of Section 4.2.2. The output will be a collection of at most $n \sum_{i=1}^{m} |T_i| + 1$ corners output by the corner oracle. These will be simple virtual VCG allocation rules, whose weight functions we denote by $\{f_i\}_{i \in [m]}^{(1)}, \ldots, \{f_i\}_{i \in [m]}^{(n \sum_{i=1}^{m} |T_i| + 1)}$. We also denote by p_j the probability placed on $VVCG\left(\{f_i\}_{i \in [m]}^{(j)}\right)$ in the output decomposition.

3: Output the allocation rule M that runs $VVCG\left(\{f_i\}_{i \in [m]}^{(j)}\right)$ with probability p_j.

Proof of Theorem 13: It follows from the correctness of the decomposition algorithm in Section 2.8 that the output allocation rule M implements the input reduced form $\vec{\pi}'$ when bidders are sampled from $\mathcal{D}'$. Now it follows from Corollary 7 that with probability at least $1 - e^{-O(n \sum_{i=1}^{m} |T_i|/\epsilon)}$ (see the proof of Theorem 12 for why the probability guaranteed by Corollary 7 is at least this large given our choice of parameters) it holds that $|\vec{\pi} - \vec{\pi}'|_\infty \leq \epsilon$ (again see the proof of Theorem 12 for why the guaranteed distance is at most ϵ given our choice of parameters). The bound on the running time follows directly from Corollary 16 and our choice of parameters. $\square$

4.5 Characterization for Correlated Biddders

Here we provide the analogue of Theorem 10 for correlated bidders. We begin by observing that, in fact, Theorem 10 already holds for correlated bidders. The reduced form is still well-defined, and nothing about the proofs in Section 4.1 requires independence across bidders. What's wrong is that the information contained in the reduced form is not sufficient for correlated bidders. Indeed, for independent bidders, the information contained in the reduced form is sufficient to verify both feasibility (due to Section 4.2.1) and bayesian incentive compatibility. For correlated bidders, while feasibility can still be verified, bayesian incentive compatibility cannot. This is because in order for bidder i to decide whether she wishes to report type A or type B when her true type is B, she needs to know the probability of receiving each item if she reports A, conditioned on the fact that the remaining bidders are sampled according to the conditional distribution induced by $t_i = B$. This information is simply not contained in the reduced form. To cope with this issue, we first propose an extended definition of reduced form for the case of correlated bidders, and a proper analogue of a virtual VCG allocation rule.

Definition 9. *A **second-order reduced form** is a vector valued function $\pi(\cdot)$ such that $\pi_{ij}(A, B)$ denotes the probability that bidder i receives item j when reporting type A, where the probability is taken over the randomness of the mechanism and the other bidders' types, assuming they are sampled from $\mathcal{D}_{-i}(B)$ and bid truthfully.*

Definition 10. *A **second-order VCG allocation rule** is defined by a collection of second-order weight functions $w_{ij} : T_i \times T_i \to \mathbb{R}$. w_{ij} maps a reported type of bidder i and true type of bidder i to a second-order bid for item j. On any profile $\vec{v}$, the second-order VCG allocation rule with weights $\vec{w}$ (denoted $SOVCG_{\mathcal{F}}(\vec{w})$) on input $\vec{v}$ selects the max-weight feasible allocation using the weights:*

$$f_{ij}(\vec{v}) = \sum_{B \in T_i} w_{ij}(\vec{v}_i, B) \Pr[\vec{v}_{-i} \leftarrow \mathcal{D}_{-i}(B)]$$

We say that a second-order VCG allocation rule with weights $\vec{w}$ is simple, if on every profile $\vec{v}$, there is a unique max-weight allocation, where the weight for allocating item j to bidder i is $f_{ij}(\vec{v})$. We now quickly observe a connection between second-order VCG allocation rules and virtual VCG allocation rules when bidders are independent, and follow with a statement of the analogue of Theorem 10 for second-order reduced forms.

Observation 4. *If bidders are independent, then for any second-order weight vector $\vec{w}$ and virtual weight vector $\vec{w}'$ with $w'_{ij}(A) = \sum_{B \in T_i} w_{ij}(A, B)$, the allocation rules $VVCG_{\mathcal{F}}(\vec{w}')$ and $SOVCG_{\mathcal{F}}(\vec{w})$ are identical for all $\mathcal{F}$.*

Proof. When bidders are independent, $\Pr[\vec{v}_{-i} \leftarrow \mathcal{D}_{-i}(B)] = \Pr[\vec{v}_{-i} \leftarrow \mathcal{D}_{-i}]$ for all B. In addition, $\Pr[\vec{v}_{-i} \leftarrow \mathcal{D}_{-i}] = \Pr[\vec{v} \leftarrow \mathcal{D}] / \Pr[t_i = \vec{v}_i]$. Therefore, the weight f_{ij} used by $SOVCG_{\mathcal{F}}(\vec{w})$ on bid vector $\vec{v}$ is just:

$$f_{ij}(\vec{v}) = \frac{\Pr[\vec{v} \leftarrow \mathcal{D}]}{\Pr[t_i = \vec{v}_i]} \sum_{B \in T_i} w_{ij}(\vec{v}_i, B)$$

The weight f'_{ij} used by $VVCG_{\mathcal{F}}(\vec{w}')$ is:

$$\begin{aligned} f'_{ij} &= w'_{ij}(\vec{v}_i) / \Pr[t_i = \vec{v}_i] \\ &= \frac{1}{\Pr[t_i = \vec{v}_i]} \sum_{B \in T_i} w_{ij}(\vec{v}_i, B) \\ &= \frac{f_{ij}(\vec{v})}{\Pr[\vec{v} \leftarrow \mathcal{D}]} \end{aligned}$$

So the weights used by $VVCG_{\mathcal{F}}(\vec{w}')$ are proportional to the weights used by $SOVCG_{\mathcal{F}}(\vec{w})$ and they will choose the same allocation on every profile. □

Theorem 15. *Let $\mathcal{F}$ be any set system of feasibility constraints, and $\mathcal{D}$ any arbitrarily correlated distribution over consumer types with finite support. Then every feasible second-order reduced form (with respect to $\mathcal{F}$ and $\mathcal{D}$) can be implemented by a distribution over at most $\sum_{i=1}^{m} |T_i|^2 + 1$ simple second-order VCG allocation rules.*

The proof of Theorem 15 parallels that of Theorem 10. We begin by observing that Observation 3 and Proposition 5 also hold in this setting and the proofs are identical. We denote the polytope of feasible second-order reduced forms with respect to $\mathcal{F}$ and $\mathcal{D}$ by $SO(\mathcal{F}, \mathcal{D})$. We now characterize the corners of $SO(\mathcal{F}, \mathcal{D})$, beginning with an analogue of Lemma 3:

Lemma 11. *Let $\vec{\pi}$ be the second-order reduced form of $SOVCG_{\mathcal{F}}(\vec{w})$ with respect to $\mathcal{D}$. Then for all $\vec{\pi}' \in SO(\mathcal{F}, \mathcal{D})$:*

$$\vec{\pi} \cdot \vec{w} \geq \vec{\pi}' \cdot \vec{w}$$

Proof. Consider any allocation rule M with second-order reduced form $\vec{\pi}''$ and denote by $M_{ij}(\vec{v})$ the probability that M awards item j to bidder i on profile $\vec{v}$. Then we can expand $\vec{\pi}'' \cdot \vec{w}$ as:

$$\begin{aligned}
\vec{\pi}'' \cdot \vec{w} &= \sum_j \sum_i \sum_{A \in T_i} \sum_{B \in T_i} w_{ij}(A, B) \pi''_{ij}(A, B) \\
&= \sum_j \sum_i \sum_{A \in T_i} \sum_{B \in T_i} w_{ij}(A, B) \sum_{\vec{v}_{-i}} \Pr[\vec{v}_{-i} \leftarrow \mathcal{D}_{-i}(B)] M_{ij}(\vec{v}_{-i}; A) \\
&= \sum_{\vec{v}} \sum_i \sum_j M_{ij}(\vec{v}) \cdot \sum_{B \in T_i} w_{ij}(\vec{v}_i, B) \Pr[\vec{v}_{-i} \leftarrow \mathcal{D}_{-i}(B)] \\
&= \sum_{\vec{v}} \sum_i \sum_j M_{ij}(\vec{v}) \cdot f_{ij}(\vec{v})
\end{aligned}$$

The second line is derived by simply exanding $\pi''_{ij}(A, B)$. The third line is derived by determining the coefficient for each $M_{ij}(\vec{v})$ in the previous line. The final line is derived by replacing $\sum_{B \in T_i} w_{ij}(\vec{v}_i, B) \Pr[\vec{v}_{-i} \leftarrow \mathcal{D}_{-i}(B)]$ with $f_{ij}(\vec{v})$. One should interpret $f_{ij}(\vec{v})$ to be the weight of awarding item j to bidder i on profile $\vec{v}$. Therefore, the allocation rule whose second-order reduced form maximizes $\vec{\pi}'' \cdot \vec{w}$ over all feasible second order reduced forms is simply the one that selects the max-weight allocation on every profile, where the weight of awarding bidder i item j on profile $\vec{v}$ is $f_{ij}(\vec{v})$. This is exactly the allocation rule $SOVCG_{\mathcal{F}}(\vec{w})$. □

Proposition 7. *Every corner in $SO(\mathcal{F}, \mathcal{D})$ can be implemented by a simple second-order VCG allocation rule, and the reduced form of any second-order VCG allocation rule is a corner in $SO(\mathcal{F}, \mathcal{D})$.*

Proof. The proof is truly identical to that of Proposition 6 after replacing Lemma 3 with Lemma 11. □

Proof of Theorem 15: Again, the proof is identical to that of Theorem 10 after replacing Proposition 6 with Proposition 7. □

We conclude this section with a discussion on the content of Theorems 10 and 15. Again, we are *not* claiming that every allocation rule can be implemented as a distribution over second-order VCG allocation rules. This is again not true, and the same example from Section 4.1 bears witness. Let's take a step back and view virtual and second-order VCG allocation rules as special cases of a more generic type of allocation rule:

Definition 11. *A **weight-scaling** allocation rule is defined by a dimension, k, and a collection of functions $\{V_{ij}, W_{ij}\}_{(i,j)\in[m]\times[n]}$. V_{ij} maps a type of bidder i to a k-dimensional weight vector ($T_i \to \mathbb{R}^k$) and W_{ij} maps the remaining types to a k-dimensional scaling vector ($\times_{i'\neq i} T_{i'} \to \mathbb{R}^k$). On any profile $\vec{v}$, the weight-scaling allocation rule with functions $\{V_{ij}, W_{ij}\}_{i,j}$ selects the max-weight allocation with weights:*

$$f_{ij}(\vec{v}) = V_{ij}(\vec{v}_i) \cdot W_{ij}(\vec{v}_{-i})$$

In other words, the weight of awarding bidder i item j is the dot product of two vectors, one contributed by bidder i's type, and the other contributed by the rest of the profile. It is not hard to see that every deterministic allocation rule can be implemented by a weight-scaling allocation rule of dimension $\prod_i |T_i|$. [3] It is also not hard

[3]Specifically, to implement any deterministic allocation rule M, index the possible profiles as $P_1, \ldots, P_k$. If on profile P_a, $t_i \neq \vec{v}_i$, set $(V_{ij})_a(\vec{v}_i) = 0$ for all j. Similarly, if $t_{-i} \neq \vec{v}_{-i}$, set $(W_{ij})_a(\vec{v}_{-i}) = 0$ for all j. If $t_{-i} = \vec{v}_{-i}$ on profile P_a, set $(W_{ij})_a(\vec{v}_{-i}) = 1$. If $t_i = \vec{v}_i$, and M awards bidder i item j on profile P_a, set $(V_{ij})_a(\vec{v}_i) = 1$. If $t_i = \vec{v}_i$ and M doesn't award bidder i item j on profile P_a, set $(V_{ij})_a(\vec{v}_i) = -1$. Then on any profile, we will have $f_{ij}(\vec{v}) = 1$ iff M awards bidder i item j on $\vec{v}$, and -1 otherwise.

to imagine that in order to specify arbitrary deterministic mechanisms as a weight-scaling allocation rule, dimension $\prod_i |T_i|$ might be necessary. However, virtual VCG allocation rules are weight-scaling allocation rules of dimension 1 (and furthermore, $W_{ij}(\vec{v}_{-i}) = 1$ for all $i, \vec{v}_{-i}$), and second-order VCG allocation rules are weight-scaling allocation rules of dimension $\max_i |T_i|$. By restricting ourselves to only care about the reduced form or second-order reduced form, we have drastically simplified the space of allocation rules. Our characterization theorems show that every allocation rule has the same reduced form as a distribution over weight-scaling allocation rule of dimension 1, and the same second-order reduced form as a distribution over weight-scaling allocation rule of dimension $\max_i |T_i|$.

Chapter 5

Revenue-Optimal Mechanisms

In this chapter, we describe how to use the results of Section 4.3 to obtain computationally efficient nearly-optimal solutions to MDMDP using only black box access to an implementation of the VCG allocation rule. As our notable contribution to obtain these results is the techniques of Sections 4.1 through 4.3, we only state our results here.

In Section 5.2, we provide a high-level overview of how to combine our results with the LPs of [DW12] to solve the MDMDP, followed by proofs. In all theorem statements, the allocation rule of the mechanism output by our algorithm is a distribution over simple virtual VCG allocation rules. There is no special structure in the pricing rule, it is just the output of a linear program. As usual, we denote by $A_{\mathcal{F}}$ an algorithm that implements the VCG allocation rule with feasibility constraints $\mathcal{F}$, and denote by $rt_{\mathcal{F}}(b)$ the runtime of $A_{\mathcal{F}}$ when each input weight has bit complexity b. We note that the mechanisms output by the following theorems can be made interim or ex-post individually rational without any difference in revenue. We are also able to accommodate bidders with hard budget constraints in our solutions. The proofs presented in Section 5.2 provide interim individually rational mechanisms without budget constraints. Ex-post individual rationality and budgets are discussed in Section 5.3.

5.1 Revenue-Maximizing Mechanisms

We first present a result for a special case of MDMDP where the feasibility is simply that no item should be given out more than once. As there is no constraint across items, to verify the feasibility of a reduced form auction, it suffices to check the corresponding single-item reduced form auction for each item separately. Thus, for this special case we can use the exact separation oracle and decomposition algorithm in Chapter 3, the end result is an exact revenue-optimal mechanism.

Theorem 16. *When the feasibility constant $\mathcal{F}$ is that no item should be given out more than once, for all $\mathcal{D}$ of finite support in $[0,1]^{nm}$ there is an exact solution for MDMDP with running time polynomial in $n, m, \sum_i |T_i|$ and ℓ, 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 BIC.*

Now, we are ready to state our main result – a solution to MDMDP with an arbitrary feasibility constraint.

Theorem 17. *For all $\epsilon, \eta > 0$, all $\mathcal{D}$ of finite support in $[0,1]^{nm}$, and all $\mathcal{F}$, given $\mathcal{D}$ and black box access to $A_{\mathcal{F}}$ there is an additive FPRAS for MDMDP. In particular, the FPRAS obtains expected revenue $OPT - \epsilon$, with probability at least $1-\eta$, in time polynomial in ℓ, $m, n, \max_{i\in[m]}\{|T_i|\}, 1/\epsilon, \log(1/\eta)$ and $rt_{\mathcal{F}}(poly(n\sum_{i=1}^{m}|T_i|, \log\log(1/\eta), \log 1/\epsilon, \ell))$, 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, its allocation rule is a distribution over simple virtual VCG allocation rules, and it can be implemented in the afore-stated running time.*

Theorem 18. *For all $\epsilon, \eta > 0$, all item-symmetric $\mathcal{D}$ of finite support in $[0,1]^{nm}$, and all item-symmetric $\mathcal{F}$,[1] given $\mathcal{D}$ and black box access to $A_{\mathcal{F}}$, there is an additive*

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

FPRAS for MDMDP. The FPRAS obtains expected revenue $OPT-\epsilon$, with probability at least $1-\eta$, in time polynomial in $\ell, m, n^c, 1/\epsilon, \log 1/\eta$ and $rt_{\mathcal{F}}(poly(n^c m, \log\log(1/\eta), \log 1/\epsilon, \ell))$, where $c = \max_{i,j} |\mathcal{D}_{ij}|$, where $|\mathcal{D}_{ij}|$ 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 17. The output mechanism is ϵ-BIC, its allocation rule is a distribution over simple virtual VCG allocation rules, and it can be implemented in the afore-stated running time.

Theorem 19. *For all $\epsilon, \eta, \delta > 0$, all item-symmetric $\mathcal{D}$ supported on $[0,1]^{nm}$ and all item-symmetric $\mathcal{F}$, given $\mathcal{D}$ and black box access to $A_{\mathcal{F}}$, there is an additive bi-criterion PRAS algorithm for MDMDP with the following guarantee: If C is the maximum number of items that are allowed to be allocated simultaneously by $\mathcal{F}$, the algorithm obtains expected revenue $OPT - (\sqrt{\epsilon} + \sqrt{\delta})C$, with probability $1-\eta$, in time polynomial in $m, n^{1/\delta}, 1/\epsilon, \log(1/\eta)$, and $rt_{\mathcal{F}}(poly(n^{1/\delta} m, \log 1/\epsilon, \log\log 1/\eta))$. In particular, the runtime does* not *depend on $|\mathcal{D}|$ at all). The output mechanism is ϵ-BIC, and can be implemented in the afore-stated running time.*

Remark 2. *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 19 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.*

5.2 Discussion and Proofs from Section 5.1

Approach. In [DW12], linear programs are provided that exactly solve MDMDP in cases with finite support (and less general feasibility constraints, namely each bidder has an upper bound on the number of items she wants to receive, and every item should be allocated to at most one bidder). However, the proposed LPs maintain variables for every type profile P, denoting the probability that bidder i receives item j on profile P, resulting in LP size proportional to $|\mathcal{D}|$. We observe that these LPs can be made more efficient by making use of the reduced form, even if just a separation oracle is provided for the feasibility of the reduced form. Indeed, the reduced form of a mechanism contains sufficient information to verify truthfulness (given the additivity of the bidders), and a separation oracle for the feasibility of the reduced form is sufficient to optimize the expected revenue of the mechanism by solving an LP. Algorithm 5 and Figure 5-1 provide the details of how we apply this approach in our setting, culminating in a proof of Theorem 17. Simply put, we use the LP approach in the following way: (a) we use a separation oracle for the proxy polytope of feasible reduced forms $F(\mathcal{F}, \mathcal{D}')$, obtained in Section 4.3, rather than the real polytope $F(\mathcal{F}, \mathcal{D})$; (b) still, we compute expected revenue for bidders sampled from the real distribution $\mathcal{D}$. For Theorem 16, the proof is basically the same except that we can solve the LP on the real polytope as we have an efficient exact separation oracle and a decomposition algorithm.

Algorithm 5 FPRAS for solving MDMDP when $\mathcal{D}$ has finite support.

1: Input: $\mathcal{D}$, $\mathcal{F}$, ϵ.
2: Set $\delta = \frac{\epsilon}{2m}$. Run the pre-processing algorithm (Algorithm 3) on input $\mathcal{D}$, with accuracy $\delta/2n$. Call the output distribution $\mathcal{D}'$.
3: Let $SO(\vec{\pi})$ be the separation oracle that on input $\vec{\pi}$ executes the separation oracle of Section 4.2.1 on input $\vec{\pi}$ for distribution $\mathcal{D}'$ and feasibility constraints $\mathcal{F}$.
4: Using SO, solve the Linear Program of Figure 5-1. Store the output as $\vec{\pi}, \vec{p}$.
5: Run the decomposition algorithm (Algorithm 4) with input $\mathcal{F}, \mathcal{D}', \vec{\pi}$. Store the output as M'. M' is a distribution over at most $n \sum_{i=1}^{m} |T_i| + 1$ simple virtual VCG allocations.
6: Output the allocation rule M' and pricing rule $\vec{p} - \delta \cdot \vec{1}$ (i.e. when bidder i reports type A, charge her $p_i(A) - \delta$).

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:

- $\vec{\pi}_i(\vec{v}_i) \cdot \vec{v}_i - p_i(\vec{v}_i) \geq \vec{\pi}_i(\vec{w}_i) \cdot \vec{v}_i - p_i(\vec{w}_i) - \delta$, 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 δ-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.
- $SO(\vec{\pi}) =$"yes," guaranteeing that the reduced form $\vec{\pi}$ is in $F(\mathcal{F}, \mathcal{D}')$.

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 5-1: A linear programming formulation for MDMDP.

Proof of Theorem 17: We use Algorithm 5. Using the additivity of the bidders, it follows that Step 4 of the algorithm outputs a reduced form/pricing rule pair $(\vec{\pi}, \vec{p})$ that is revenue-optimal with respect to all δ-BIC, IR reduced form/pricing rule pairs, except that the reduced forms that are searched over belong to $F(\mathcal{F}, \mathcal{D}')$ and may be infeasible with respect to $\mathcal{D}$. We proceed to argue that the achieved revenue is nearly-optimal with respect to all BIC, IR reduced form/pricing rule pairs for which the reduced form lies inside $F(\mathcal{F}, \mathcal{D})$. So let $(\vec{\pi}^*, \vec{p}^*)$ denote an optimal such pair, and let OPT denote its expected revenue. By Theorem 12, we know that, with high probability, there is some reduced form $\vec{\pi}' \in F(\mathcal{F}, \mathcal{D}')$ satisfying $|\vec{\pi}' - \vec{\pi}^*|_\infty \leq \delta/2n$. So, if we let $\vec{p}' = \vec{p}^* - \frac{\delta}{2} \cdot \vec{1}$, it is obvious that the reduced form $(\vec{\pi}', \vec{p}')$ is δ-BIC. It is also obvious that it is individually rational. Finally, it is clear that value of the LP achieved by $(\vec{\pi}', \vec{p}')$ is exactly $m\delta/2 = \epsilon/4$ less than the value of $(\vec{\pi}^*, \vec{p}^*)$. So because $(\vec{\pi}', \vec{p}')$ is in the feasible region of the LP of Figure 5-1, the reduced form/price

rule pair output by Step 4 of Algorithm 5 has expected revenue at least OPT $- \epsilon/4$. Noticing that we subtract an additional δ from the price charged to each bidder in Step 6 of the algorithm, we get that the end price rule makes expected revenue at least OPT $- \epsilon$.

We argue next that the mechanism output by Algorithm 5 is ϵ-BIC and IR. To see this let M' be the allocation rule (computed in Step 5 of the algorithm), which implements the reduced form $\vec{\pi}$ (computed in Step 4) with respect to $\mathcal{D}'$. Let also $\vec{\pi}'$ denote the reduced form of M' with respect to $\mathcal{D}$. By Theorem 13, we know that, with high probability, $|\vec{\pi} - \vec{\pi}'|_\infty \leq \delta/2n$. Therefore, given that $(\vec{\pi}, \vec{p})$ is δ-BIC, we immediately get that $(\vec{\pi}', \vec{p} - \delta \cdot \vec{1})$ is 2δ-BIC. So allocation rule M' with pricing rule $\vec{p} - \delta \cdot \vec{1}$ comprises an ϵ-BIC mechanism. Also because $(\vec{\pi}, \vec{p})$ is IR and $|\vec{\pi} - \vec{\pi}'|_\infty \leq \delta/2n$, we immediately get that $(\vec{\pi}', \vec{p} - \delta \cdot \vec{1})$ is IR, and hence that allocation rule M' with pricing rule $\vec{p} - \delta \cdot \vec{1}$ is IR.

Overall the above imply that the allocation rule M' and the pricing rule $\vec{p} - \delta \cdot \vec{1}$ output in Step 6 of our algorithm comprise an ϵ-BIC and IR mechanism, whose pricing rule achieves revenue at least $OPT - \epsilon$. Moreover, it is immediate from Theorems 12 and 13 that Algorithm 5 runs in time polynomial in c, n, $\sum_{i=1}^{m} |T_i|$, $1/\epsilon$, and $rt_{\mathcal{F}}(\text{poly}(n \sum_{i=1}^{m} |T_i|, \log 1/\epsilon, c))$, where c is as in the statement of the theorem.

Finally, the way we chose our parameters in Algorithm 5 the probability of failure of the algorithm is $1 - e^{-\Omega(mn^2 \sum_i |T_i|/\epsilon)}$ by Theorems 12 and 13. Trading off probability of error with ϵ (as per Remark 1) we complete the proof of Theorem 17.$\square$

Theorems 18 and 19 are obtained by combining Theorem 17 with tools developed in [DW12]. Both theorems are based on the following observation, generalizing Theorem 2 of [DW12]: If $\mathcal{D}$ is item-symmetric and the feasibility constraints $\mathcal{F}$ are also item-symmetric then there exists an optimal mechanism that is:

1. item-symmetric, i.e. for all bidders i, all types $\vec{v}_i \in T_i$, and all item-permutations σ, the reduced form of the mechanism satisfies $\vec{\pi}_i(\sigma(\vec{v}_i)) = \sigma(\vec{\pi}_i(\vec{v}_i))$; this means that the reduced form on a permuted type of a bidder is the same permutation of the reduced form of the un-permuted type of the bidder.

2. strongly-monotone, i.e. for all bidders i, and items j and j', $v_{ij} \geq v_{ij'} \implies \pi_{ij}(\vec{v}_i) \geq \pi_{ij'}(\vec{v}_i)$.

Using this structural observation for optimal mechanisms, we sketch the proofs of Theorems 18 and 19.

Proof of Theorem 18: (Sketch) Given our structural observation for optimal mechanisms, we can—without loss of generality—rewrite the LP of Figure 5-1, while at the same time enforcing the above constraints, i.e. searching over the set of item-symmetric, strongly-monotone reduced forms. Indeed, to save on computation we can write a succinct LP on variables $\{\pi_{ij}(\vec{v}_i)\}_{i,j,\vec{v}_i \in E_i}$ where, for every bidder i, E_i is a sufficient (based on the above symmetries) set of representative types, e.g. we can take $E_i = \{\vec{v}_i \mid v_{i1} \geq \ldots \geq v_{in}\}$. We refer the reader to [CDW12a] for the explicit form of the succinct LP. The benefit of the succinct formulation is that $|E_i| \leq n^c$, where c is as in the statement of Theorem 18, so the size of the succinct LP is polynomial in m, n^c and ℓ, where ℓ is as in the statement of the theorem.

But we also need to come up with an efficient separation oracle for our setting. Checking violation of the strong-monotonicity property is easy to do in time linear in $O(mn)$ and the description of $\{\pi_{ij}(\vec{v}_i)\}_{i,j,\vec{v}_i \in E_i}$. So it remains to describe a separation oracle determining the feasibility of a succinct description $\{\pi_{ij}(\vec{v}_i)\}_{i,j,\vec{v}_i \in E_i}$ of an item-symmetric reduced form. One approach to this would be to expand out the succinct description of the item-symmetric reduced form to a full-fledged reduced-form and invoke the separation oracle developed in Sections 4.2 through 4.3. However, this would make us pay computation time polynomial in $\sum_i |T_i|$, and the whole point of using item-symmetries is to avoid this cost. To circumvent this, we take the following approach:

- First, let $F_S(\mathcal{F}, \mathcal{D})$ be the set of item-symmetric reduced forms that are feasible with respect to $\mathcal{F}$ and $\mathcal{D}$;
- $F_S(\mathcal{F}, \mathcal{D})$ is a polytope, as it is the intersection of the polytope $F(\mathcal{F}, \mathcal{D})$ and the item-symmetry constraints; moreover, every point in $F_S(\mathcal{F}, \mathcal{D})$ has a succinct description of the form $\{\pi_{ij}(\vec{v}_i)\}_{i,j,\vec{v}_i \in E_i}$ where, for all i, E_i is defined as above;

- What are the corners of $F_S(\mathcal{F}, \mathcal{D})$? These can be implemented by item-symmetric virtual VCG allocation rules whose weight-functions are item-symmetric. The proof of this is identical to the proof of Proposition 6 noticing that $F_S(\mathcal{F}, \mathcal{D})$ lies in the lower-dimensional space spanned by the item-symmetries. We note that we do not require the virtual VCG allocation rules to be *simple* in the same sense defined in Section 4.1, as this could violate item-symmetry.

- However, given an item-symmetric weight vector $\vec{w}$ how do we even run an item-symmetric virtual VCG allocation rule corresponding to $\vec{w}$? There are two issues with this: (a) how to enforce the item-symmetry of the virtual VCG allocation rule; and (b) there could be multiple item-symmetric virtual VCG allocation rules consistent with $\vec{w}$, e.g., if $\vec{w}$ is perpendicular to a facet of $F_S(\mathcal{F}, \mathcal{D})$. Here is how we resolve these issues: First, we apply Lemma 4 to the symmetric weight vector $\vec{w}$ to get a non-symmetric weight vector $\vec{w}'$ (we may do this transformation explicitly or do a lazy-evaluation of it—this is relevant only for our computational results three bullets down). When bidders submit their types, we pick a permutation σ uniformly at random, and permute the names of the items. Then we use the simple virtual VCG allocation rule corresponding to $\vec{w}'$. Finally, we un-permute the names of the items in the allocation. We denote this allocation rule by $S.VVCG_{\mathcal{F}}(\vec{w})$. It is clear that $S.VVCG_{\mathcal{F}}(\vec{w})$ is well-defined (i.e. no tie-breaking will ever be required), is item-symmetric, and defines a virtual VCG allocation rule w.r.t. the original weight vector $\vec{w}$. Given the above discussion, every item-symmetric weight vector $\vec{w}$ has a succinct description of the form $\{w_{ij}(\vec{v}_i)\}_{i,j,\vec{v}_i \in E_i}$, which defines uniquely an item-symmetric virtual VCG allocation rule w.r.t. $\vec{w}$ (namely, $S.VVCG_{\mathcal{F}}(\vec{w})$);

- Given the above definitions and interpretations, we can generalize the results of Sections 4.1 and 4.2 to the polytope $F_S(\mathcal{F}, \mathcal{D})$.

- Next we discuss how to extend the computationally-friendly results of Section 4.3 to the item-symmetric setting, while maintaining the computational complexity of all algorithms polynomial in m, $\max_i |E_i| = O(n^c)$ and ℓ, where

ℓ is as in the statement of the theorem. We define an item-symmetric distribution $\mathcal{D}'$ as follows: We draw $k'' = k + k' \sum_{i=1}^{m} |E_i|$ profiles of bidders $P_1, \ldots, P_{k''}$ from $\mathcal{D}$ as in Section 4.3, except that, for each bidder i, we draw k' profiles conditioning on the type of the bidder being each element of E_i and not T_i. Then we define (without explicitly writing down) $\mathcal{D}'$ to be the two-stage distribution that in the first stage draws a random profile from $P_1, \ldots, P_{k''}$ and in the second-stage permutes the items using a uniformly random item-permutation. We claim that using $T = \text{poly}(mn^c)$ in Algorithm 3 suffices to obtain an analog of Theorems 12 and 13 for our setting with probability of success $1 - e^{-1/\epsilon}$. (We address the running time shortly.) The reason we can save on the number of samples is that we are working with $F_S(\mathcal{F}, \mathcal{D})$ and hence all reduced forms are forced to be item-symmetric. So we need to prove concentration of measure for a smaller number of marginal allocation probabilities.

- Unfortunately, we cannot afford to compute reduced forms with respect to $\mathcal{D}'$, as we can't in general evaluate the reduced form of $S.VVCG_{\mathcal{F}}(\vec{w})$ on a given type profile without making prohibitively many queries to $A_{\mathcal{F}}$. Instead, we will also independently sample item permutations $\sigma_1, \ldots, \sigma_{k''}$, and associate the permutation σ_α with P_α in the following sense. If a given type profile was sampled by $\mathcal{D}'$ after sampling P_α in the first stage of $\mathcal{D}'$, we will permute the items by (just) σ_α when evaluating $S.VVCG_{\mathcal{F}}(\vec{w})$ on that profile, instead of taking a uniformly random permutation. In other words, we have removed the randomness in evaluating $S.VVCG_{\mathcal{F}}(\vec{w})$ and fixed the applied item-permutation to some σ_α, which was chosen uniformly at random. Doing so, we still have the same expectations as in the previous bullet, and we can deduce that the reduced form of $S.VVCG_{\mathcal{F}}(\vec{w})$ when consumers are sampled from $\mathcal{D}$ is very close to the reduced form of $S.VVCG_{\mathcal{F}}(\vec{w})$ when consumers are sampled from $\mathcal{D}'$ (while only using item-permutation σ_α to run $S.VVCG_{\mathcal{F}}(\vec{w})$ on all profiles sampled from $\mathcal{D}'$ after sampling P_α in the first stage of $\mathcal{D}'$, as explained above).

- With the above modifications, for both Theorems 12 and 13 the running time

of the corresponding algorithm can be made polynomial in ℓ', m, n^c, $1/\epsilon$ and $rt_{\mathcal{F}}(\text{poly}(n^c, m, \log 1/\epsilon, \ell'))$, where ℓ' is the max of ℓ (see statement) and the bit complexity of the coordinates of the input to the algorithms. To achieve this we only do computations with succinct descriptions of item-symmetric reduced forms and weight vectors. What we need to justify further is that we can do exact computations on these objects with respect to the distribution $\mathcal{D}'$ given oracle access to $A_{\mathcal{F}}$ in the afore-stated running time. For this it suffices to be able compute the succinct description of the reduced form $\vec{\pi}$ of $S.VVCG_{\mathcal{F}}(\vec{w})$ (using permutation σ_α on profiles coming from P_α as explained above). The small obstacle is that the support of $\mathcal{D}'$ is not polynomial in the required running time, but it suffices to do the following. For each profile P_α find the allocation output by the (non item-symmetric) virtual VCG allocation rule corresponding to the perturbed vector $\vec{w}'$, after relabeling the items according to σ_α. This we can do in the allotted running time with a lazy evaluation of the perturbation. Then, for all i, $\vec{v}_i$ and j, to compute $\pi_{ij}(\vec{v}_i)$ do the following: for all profiles P_α, let $x_\alpha(i, \vec{v}_i)$ denote the number of τ such that $\tau(\vec{v}_i)$ matches the type $t_i(\alpha)$ of bidder i in P_α. Let $y_\alpha(i, \vec{v}_i)$ denote the number of τ such that $\tau(\vec{v}_i)$ matches the type of bidder i in P_α, and item $\tau(j)$ is awarded to bidder i. It is easy to compute $x_\alpha(i, \vec{v}_i)$: let $J_v^1 = \{j | v_{ij} = v\}$, and J_v^2 be the set of items that bidder i values at v in profile P_α. Then if $|J_v^1| \neq |J_v^2|$ for any v, $x_\alpha(i, \vec{v}_i) = 0$. Otherwise, $x_\alpha(i, \vec{v}_i) = \prod_v |J_v^1|!$, because $\tau(\vec{v}_i)$ matches the type of bidder i iff τ maps all of J_v^1 to J_v^2 for all v. Computing $y_\alpha(i, \vec{v}_i)$ is also easy: simply break up J_v^2 into two sets: $J_v^2(W)$ of items that bidder i values at v and wins, and $J_v^2(L)$ of items that bidder i values at v and loses. Then if $x_\alpha(i, \vec{v}_i) \neq 0$, $y_\alpha(i, \vec{v}_i) = |J_{v_{ij}}^2(W)| \cdot (|J_{v_{ij}}^2| - 1)! \cdot \prod_{v \neq v_{ij}} |J_v^2|!$, because bidder i is awarded item $\tau(j)$ and $\tau(\vec{v}_i)$ matches the type of bidder i iff $\tau(j) \in J_{v_{ij}}^2(W)$, and τ maps J_v^1 to J_v^2 for all v. Once we've computed $x_\alpha(i, \vec{v}_i)$ and $y_\alpha(i, \vec{v}_i)$, it is easy to see that:

$$\pi_{ij}(\vec{v}_i) = \frac{1}{|\{\alpha | x_\alpha(i, \vec{v}_i) > 0\}|} \sum_{\alpha | x_\alpha(i, \vec{v}_i) > 0} \frac{y_\alpha(i, \vec{v}_i)}{x_\alpha(i, \vec{v}_i)}.$$

The above bullet points explain briefly how our ideas from the previous sections are modified for item-symmetric distributions. The complete details are omitted.□

Proof of Theorem 19: We combine Theorem 18 with (i) a discretization of the hypercube so that every $\vec{v}$ in the support of the distribution has $v_{ij} = k\delta$, $k \in \mathbb{N}$, for all i, j; and (ii) the approximately-BIC to BIC reduction of Section 6 of [DW12], informally stated below.

Informal Theorem 3. *(Reworded from [DW12]) Let $\mathcal{C} = \times_i \mathcal{C}_i$ and $\mathcal{C}' = \times_i \mathcal{C}'_i$ be product distributions sampling every additive bidder independently from $[0,1]^n$. Suppose that, for all i, $\mathcal{C}_i$ and $\mathcal{C}'_i$ can be coupled so that with probability 1, $\vec{v}_i$ sampled from $\mathcal{C}_i$ and $\vec{v}'_i$ sampled from $\mathcal{C}'_i$ satisfy $v_{ij} \geq v'_{ij} \geq v_{ij} - \delta$ for all j. If M' is any ϵ-BIC mechanism for $\mathcal{C}'$, then with* exact *knowledge of the reduced form of M' with respect to $\mathcal{C}'$, we can transform M' into a BIC mechanism for $\mathcal{C}$ while only losing $O(C(\sqrt{\delta} + \sqrt{\epsilon}))$ revenue, where C is the maximum number of items that are allowed to be allocated simultaneously. In item symmetric settings, the reduction runs in time polynomial in $n^{1/\delta}, m$.*

While doing the discretization is straihgtforward, there is an issue with applying the aforementioned approximately-BIC to BIC reduction. To directly apply the reduction in our setting, one might try to take $\mathcal{C}'$ to be the $\mathcal{D}'$ from the sampling procedure. Unfortunately, this doesn't work because $\mathcal{D}'$ is correlated across bidders. Instead, we might try to take $\mathcal{C}'$ to be $\mathcal{D}$. This too doesn't work because we can't exactly compute the reduced form of a mechanism with respect to $\mathcal{D}$. We can, however, compute the reduced form of a mechanism with respect to $\mathcal{D}$ with quite good accuracy. So we will prove a quick lemma about the quality of the reduction proposed in [DW12] in our setting. Virtually the same lemma is used in [HKM11] where the ideas behind this reduction originated, but in a different setting.

Lemma 12. *Let $\mathcal{C}$ and $\mathcal{C}'$ satisfy the hypotheses of Theorem 4 in [DW12], and let M' be a γ-BIC mechanism whose reduced form with respect to $\mathcal{C}'$ is $\vec{\pi}'$ (which is possibly unknown). Then with knowledge of some $\vec{\pi}$ such that $|\vec{\pi} - \vec{\pi}'|_1 \leq \epsilon$, the*

reduction of [DW12] transforms M' into a 2ϵ-BIC mechanism for $\mathcal{C}$ while only losing $O(C(\sqrt{\delta}+\sqrt{\gamma}))$ revenue, where C is as above.

Proof of Lemma 12: We avoid repeating a complete description of the reduction and refer the reader to [DW12] for more details. At a high level, the reduction is the following. The new mechanism M is a two-stage mechanism. First, each bidder i, independently from the other bidders, plays a VCG auction against make-believe replicas of herself, drawn independently from $\mathcal{C}_i$, to purchase a surrogate from a collection of surrogates drawn independently from $\mathcal{C}'_i$. (The items of the per-bidder VCG auction are the surrogates and, in particular, they have nothing to do with the items that M is selling. Moreover, the feasibility constraints of the VCG auction are just unit-demand constraints on the bidder-side and unit-supply constraints on the item-side.) After each bidder buys a surrogate, the purchased surrogates play M' against each other. Each bidder receives the items their surrogate is awarded and pays the price the surrogate pays in M', as well as a little extra in order to buy the surrogate in the per-bidder VCG auction. The truthfulness of the two-stage mechanism M boils down to the truthfulness of VCG: If we can exactly evaluate the value of bidder i of type $\vec{v}_i$ for being represented by each surrogate $\vec{s}_i$, and we use these values in computing the VCG allocation in the per-bidder VCG auction, then M is BIC. Moreover, it is shown in [DW12] that the distribution of surrogates that play M' is exactly $\mathcal{C}'$. So, this implies that, if we know exactly the reduced form of M' with respect to $\mathcal{C}'$, we can exactly compute the value of the bidder for each surrogate, and M will be BIC.

If we only know the reduced form of M' with respect to $\mathcal{C}'$ within ϵ in ℓ_1-distance, then we cannot exactly evaluate the value of bidder $\vec{v}_i$ for being represented by surrogate $\vec{s}_i$, but we can evaluate it within ϵ. Suppose that we run per-bidder VCG auctions using our estimates. In the VCG auction corresponding to bidder i, suppose $\vec{v}_i$ is the true type of the bidder, let $\vec{\pi}_1$ be our estimate of the reduced form of the surrogate that was sold to the bidder, let p_1 be the price that surrogate pays in M', and let q_1 denote the price paid for that surrogate in VCG. Let $(\vec{\pi}_2, p_2, q_2)$ be the corresponding triplet for another surrogate that the bidder would win by misreporting

her type. By the truthfulness of VCG,

$$\vec{v}_i \cdot \vec{\pi}_1 - p_1 - q_1 \geq \vec{v}_i \cdot \vec{\pi}_2 - p_2 - q_2.$$

The question is how much the bidder regrets not misreporting to the VCG auction given that the true reduced form of surrogate $\vec{s}_i$ in M' is some $\vec{\pi}'_i$, which was false-advertised in the VCG auction as $\vec{\pi}_i$, $i = 1, 2$. Given that $|\vec{\pi}'_i - \vec{\pi}_i|_1 \leq \epsilon$ and remembering that each $\vec{v}_i \in [0, 1]^n$ we get:

$$\vec{v}_i \cdot \vec{\pi}'_1 \geq \vec{v}_i \cdot \vec{\pi}_1 - \epsilon$$
$$\vec{v}_i \cdot \vec{\pi}_2 \geq \vec{v}_i \cdot \vec{\pi}'_2 - \epsilon.$$

Therefore,

$$\vec{v}_i \cdot \vec{\pi}'_1 - p_1 - q_1 \geq \vec{v}_i \cdot \vec{\pi}'_2 - p_2 - q_2 - 2\epsilon.$$

This means that if bidder i were to misreport her type to get surrogate $\vec{s}_2$, her true utility would increase by at most 2ϵ. So M is 2ϵ-BIC.

Finally, we can use the same argument as in [DW12] to show that the reduction loses at most $O(C(\sqrt{\delta} + \sqrt{\gamma}))$ in revenue.□

Coming back to the proof of Theorem 19, if we just discretized the value distribution without running the approximately-BIC to BIC reduction of Lemma 12, we would get a mechanism that is $(\epsilon+\delta)$-BIC and suboptimal by $(\epsilon+\delta)C$ (Lemma 3 of [DW12] pins down the loss of truthfulness/revenue due to discretization of the value distribution in multiples of δ, and Theorem 18 bounds the additional loss due to computational constraints). If we apply the reduction of Lemma 12 on this mechanism, we can turn it into one that is ϵ-BIC and suboptimal by $O(\sqrt{\epsilon} + \sqrt{\delta})C$. The reason we even bother running the approximately-BIC to BIC reduction when we don't get a truly BIC mechanism in the end is because, while keeping our algorithm efficient, ϵ can be made as small as $1/\text{poly}(n, m)$, while δ needs to stay a fixed constant. So after our reduction we obtained a qualitatively stronger result, namely one whose distance from truthfulness can be made arbitrarily small in polynomial time. □

5.3 Accommodating Budget Constraints

In this section we show a simple modification to our solutions that allows them to accommodate budget constraints as well. To do this, we simply cite an observation from [DW12]. There, it is observed that if the solution concept is interim individual rationality, then the LP that finds the revenue-optimal reduced form can be trivially modified to accommodate budget constraints. In Figure 5-2 we show how to modify our LP from Figure 5-1 to accommodate budget constraints.

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:

- $\vec{\pi}_i(\vec{v}_i) \cdot \vec{v}_i - p_i(\vec{v}_i) \geq \vec{\pi}_i(\vec{w}_i) \cdot \vec{v}_i - p_i(\vec{w}_i) - \delta$, 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 δ-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.
- $SO(\vec{\pi}) =$"yes," guaranteeing that the reduced form $\vec{\pi}$ is in $F(\mathcal{F}, \mathcal{D}')$.
- $p_i(\vec{v}_i) \leq B_i$, for all bidders i and types $\vec{v}_i \in T_i$, **guaranteeing that no bidder i pays more than their budget B_i.**

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 5-2: A linear programming formulation for MDMDP that accommodates budget constraints.

It is also shown in [DW12] that accommodating budget constraints comes at a cost. First, it is shown that without budget constraints, one can turn any interim-IR mechanism into an ex-post IR mechanism with no loss in revenue. However,

with budget constraints, there is a potentially large gap between the revenue of the optimal ex-post IR mechanism and the optimal interim IR mechanism. In other words, accommodating budget constraints requires accepting interim IR instead of ex-post IR. Second, the approximately-BIC to BIC reduction of [DW12] (used in the proof of Theorem 22) does not respect budget constraints. So to accommodate budgets in Theorem 22 the output mechanism needs to be δ-BIC instead of ϵ-BIC.[2] Relative to the ability to naturally accommodate budget constraints, these costs are minor, but we state them in order to correctly quantify the settings our techniques solve.

[2]Recall that the runtime required to find and execute the mechanism of Theorem 22 is polynomial in $1/\epsilon$ but exponential in $1/\delta$.

Chapter 6

Approximately Optimal Mechanisms

6.1 Overview of Our Results

In Chapter 4 and 5, we have 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 $\mathcal{F}$. 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 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 chapter 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 12. *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 func-*

tions $f_1, \ldots, f_m$, 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 20 of Section 6.6.

Informal Theorem 4. *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 6.6 extensions for distributions of infinite support and improved runtimes in certain cases, making use of techniques from [DW12]. 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 12) 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

[1]The following simple black-box transformation achieves this: first zero-out all negative coordi-

this is not necessarily true 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

nates 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.

as all settings with risk-neutral bidders is at most 2^n (make a meta-item for each possible subset of items). In Section 6.9 we discuss the following observation and give examples of settings with low additive dimension, including settings where bidders have symmetric submodular valuations [BKS12].

Observation 5. *In any setting with additive dimension d, Informal Theorem 4 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.*

6.1.1 Approach and Techniques.

Our main result for this chapter, as well as the one in Chapter 5, 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 [Bor91, Bor07, CKM11], it was only recently that efficient algorithms were obtained [CDW12a, AFH$^+$12]. These approaches also generalized to serving many copies of an item with a matroid feasibility constraint on which bidders can be served an item simultaneously [AFH$^+$12], but for more general feasibility constraints there seemed to be an obstacle in even defining necessary and sufficient conditions for feasibility, let alone checking them efficiently.

In view of this difficulty, it is quite surprising that a general approach for the problem exists (Chapter 4). The main realization was that, for arbitrary feasibility

[2]The interim rule of an auction, also called the reduced form 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 2.5.

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 in Chapter 4 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 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 Chapter 4 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

in Chapter 4 with our weird separation oracle, and the decomposition algorithm with this new decomposition approach, we obtain the proof of our main theorem (Informal Theorem 4 above, and Theorem 20 in Section 6.6). Our approach is detailed in Sections 6.3, 6.4 and 6.5.

6.1.2 Previous Work

In his seminal paper, Myerson solved the single-item case of the MDMDP [Mye81]. 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. On the multi-dimensional front, the progress has been slow. Our result in Chapter 5 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 chapter 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.

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 [BKV05, BLP06, HL10, DR10, BH11, HKM11]. 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.

Weird Separation Oracle, Approximation, and Revenue Optimization.

Grötschel et al. [GLS81] 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 [Jan02]. 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 P_2 is a polytope contained in P. A precise definition of P_1 and P_2 is given in Section 6.3.1. 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 [Jan02] is not enough to accomplish this, which motivates our stronger results.

But there is another, crucial reason that prevents using the results of [Jan02], and for that matter [GLS81] (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 [Jan02] 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 [Jan02] or [GLS81] as a black box, and requires us to provide our own separation to optimization reduction, together with additional optimization tools.

6.2 Preliminaries for Weird Separation Oracle

Throughout this chapter, 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 12).

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.

6.3 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 6.3.1. 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 6.3.2 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 6.5. Our basic objects for this section are encapsulated in the following definition.

Definition 13. *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.

6.3.1 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 [GLS81]. It is not obvious, but perhaps not difficult to see also that if $\mathcal{A}$ is a maximal-in-range algorithm,[3] then $P_1 = P_2$. It turns out that in this case, WSO (as defined in Section 6.3.2) 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.

Lemma 13. $P_0 \subseteq P_1 \subseteq P_2$.

Proof. 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 $\vec{\pi} \in P_2(A, \mathcal{D})$. □

6.3.2 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 6.4 using lemmas proved here. That using WSO achieves good revenue

[3]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 $\vec{w}$, $\mathcal{A}(\vec{w}) \in \text{argmax}_{\vec{x} \in S}\{\mathcal{A}(\vec{x}) \cdot \vec{w}\}$.

begins with Fact 1. That we can implement any reduced form deemed feasible by WSO begins with Lemma 14. We define WSO in Figure 6-1.

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

- "**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 12 of Section 6.5. The only place that requires an appropriate choice of N is the proof of Lemma 14.

[b] The appropriate choice of δ for our use of WSO is provided in Lemma 16 of Section 6.5. The only place that requires an appropriate choice of δ is the proof of Lemma 14.

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

Figure 6-1: A "weird" separation oracle.

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 [GLS81] 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 $\mathcal{A}$.[4] 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. Now, we need to prove some facts about WSO despite this.

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

Fact 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, $P_1 \subseteq Q$.*

Proof. 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 $P_1 \subseteq Q$. □

Fact 2. *If $\vec{\pi} \in P_1$, then $WSO(\vec{\pi}) =$ "yes."*

Proof. 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 $\vec{\pi} \notin P_1$. □

Corollary 8. *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.*

Proof. By Fact 2, whenever WSO rejects $\vec{\pi}$, $\vec{\pi} \notin P_1$. By Fact 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 $\vec{\pi}$. □

Lemma 14. *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 $\vec{\pi} \in Conv(\{\mathcal{A}(\vec{w}) | \vec{w} \in S\})$.*

Proof. 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 $\vec{\pi} \notin Conv(\{\mathcal{A}(\vec{w}) | \vec{w} \in S\})$, then $P(S)$ is non-empty. This is because when $\vec{\pi} \notin Conv(\{\mathcal{A}(\vec{w}) | \vec{w} \in S\})$, 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 16 in Section 6.5 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 12 in Section 6.5 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 $\vec{\pi} \in Conv(\{\mathcal{A}(\vec{w}) | \vec{w} \in S\})$. $\square$

6.4 Approximately Maximizing Revenue using WSO

In this section, we show that running the revenue maximizing LP in Figure 5-1 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 in Chapter 5 is the following. We start by creating a proxy distribution $\mathcal{D}'$ that is a (correlated across bidders) uniform distribution over $\text{poly}(n, T, 1/\epsilon)$ 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 in Figure 5-1 outputs a reduced form whose revenue is at least $\text{OPT} - \epsilon$. 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 14. *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*

Given this definition, and for the same $\mathcal{D}'$ used in Chapter 5, 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 14), 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 6.3.

We continue to bound the revenue of the reduced form output by our LP of in Figure 5-1. Denote by $Rev(F)$ the revenue obtained by the LP of in Figure 5-1, and by $Rev(P_i)$ the revenue obtained by replacing P with P_i.

Lemma 15. $Rev(P_0) \geq \alpha Rev(F) \geq \alpha(OPT - \epsilon)$.

Proof. Let $(\vec{\pi}^*, \vec{p}^*)$ denote the reduced form output by the LP in Figure 5-1. 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 $\text{OPT} - \epsilon$, it is clear that revenue of the reduced form output in the new LP is at least $\alpha(\text{OPT} - \epsilon)$. □

Now, denote by $Rev(WSO)$ the revenue obtained by replacing P with WSO in Figure 5-1. 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 5-1 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,[5] 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 5-1 with P_0 in place of $F(\mathcal{F}, \mathcal{D}')$.

We now want to use Lemma 15 to lower bound $Rev(WSO)$. This is almost a direct corollary of Fact 1. The only remaining step is understanding the ellipsoid algorithm.

[5]These parameters were computed in Chapter 4 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 17 in Section 6.5.

Proposition 8. *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.*

Proof. 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 separtion oracle for Q_0 and WSO' as used in Proposition 8? 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.[6] However, Corollary 8 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. □

Corollary 9. $Rev(WSO) \geq Rev(P_0)$.

Proof. Consider running the LP of Figure 5-1 with WSO. The optimization version of the ellipsoid algorithm will do a binary search on possible values for the objective

[6] 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.

function and solve a separate feasibility subproblem for each. Proposition 8 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 $Rev(WSO) \geq Rev(P_0)$. □

Corollary 10. *$Rev(WSO) \geq \alpha(OPT - \epsilon)$.*

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 14:

Corollary 11. *Let $\vec{\pi}^*$ denote the reduced form output by the LP of Figure 5-1 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.*

Proof. Because $\vec{\pi}^*$ is output, we have $WSO(\vec{\pi}^*) =$ "yes." Lemma 14 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. □

At this point, we have shown that the reduced form $\vec{\pi}^*$ and pricing rule p^* computed by the LP of Figure 5-1 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 $\mathcal{D}'$*.

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 refer the reader

to Section 4.4.1 for a formal proof of the same fact when using $F(\mathcal{F}, \mathcal{D}')$ rather than WSO in the LP of Figure 5-1 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 6.5.

6.5 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 6.4 can be computed in polynomial time, and that both WSO from Section 6.3 and our revenue maximizing LP from Section 6.4 run in polynomial time.

Analyzing WSO from Section 6.3. We start with the appropriate choice of δ. The proof of the following lemma is in Section 6.8.1.

Lemma 16. *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 14. As Lemma 16 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 12, and proved in Section 6.8.1.

Corollary 12. *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 16.

Note that Lemma 16 and Corollary 12 complete the description of WSO, and establish the truth of Lemma 14.

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 16 and Corollary 12, we can bound the running time of WSO. This is stated below as Corollary 13 and proved in Section 6.8.1.

Corollary 13. *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$(d, \ell, b, rt_{\mathcal{A}}(\text{poly}(d, \ell, b)))$.*

Computing Reduced Forms. In Section 6.4 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 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 Section 6.8.2. 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 6.4 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, 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 6.4 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 6.4 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 13.

Corollary 14. *Let b denote the bit complexity of $\vec{\pi}$. Then on input $\vec{\pi}$, WSO terminates in time*

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

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 Section 6.8.3.

Lemma 17. *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 6.4 is both correct and computationally efficient.

[7]See what we mean by "replacing P with WSO" in Section 6.4.

6.6 Formal Theorem Statements

In this section we provide our main theorem, formalizing Informal Theorem 4. In Section 6.7, we also provide two extensions of our theorem to item-symmetric settings using the techniques of [DW12]. These extensions are Theorems 21 and 22 of Section 6.7. 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 6.4). Theorems 20, 21 and 22 follow directly from Sections 6.3 through 6.5 in the same way that their corresponding theorems (Theorems 17 through 19) in Chapter 5 follow, after replacing the separation oracle for $F(\mathcal{F}, \mathcal{D}')$ with WSO in the LP of Figure 5-1. In all theorem statements, $rt_A(x)$ denotes the runtime of algorithm A on inputs of bit complexity x.

Theorem 20. *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 20 and its extensions (Theorems 21 and 22) 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 Section 5.3.

6.7 Extensions of Theorem 20

This section contains extensions of Theorem 20 enabled by the techniques of [DW12].

Theorem 21. *For all $\epsilon, \eta > 0$, item-symmetric $\mathcal{D}$ of finite support in $[0,1]^{nm}$, 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: 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 20. The output mechanism is ϵ-BIC, and can be implemented in the same running time.*

Theorem 22. *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 3. *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 22 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.*

6.8 Omitted Proofs from Section 6.5

6.8.1 The Runtime of WSO

Proof of Lemma 16: 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 14):

$$t' \leq d$$

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

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

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 $\delta = 2^{-\text{poly}(d,\ell,b)}$. $\square$

Lemma 18. *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 16, 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 18: 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,[8] but useful for the analysis. Define $P'(S)$ by removing the first constraint.

[8]Since $P \in [-1, 1]^d$, WLOG we can also assume $\vec{\pi} \in [-1, 1]^d$, thus from the constraints of $P(S)$

That is, $P'(S)$ is the intersection of the following halfspaces (this is the same as $P'(S)$ from the proof of Lemma 16):

$$t \leq d$$

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

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

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 16 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 $(t, \vec{w})$ 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 $2^{-\text{poly}(d,\ell,b)}$. $\square$

---

it follows that $t \leq d$.

Proof of Corollary 12: By Lemma 18, 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 14), 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 13: By the choice of N in Corollary 12, 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 [GLS88], $\widehat{WSO}$ 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. □

We use the following theorem to summarize our results for WSO.

Theorem 23. *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 $\mathcal{A} \in P$ 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 the 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 $\vec{w} \in [-1,1]^d$, $rt_{\mathcal{A}}(y)$ 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 8 we know P_1 is contained in all halfspaces output by WSO, therefore $\alpha P \in P_1$ is also contained in all halfspaces. The second claim is proved in Lemma 14, and the third claim is proved in Corollary 13. □

6.8.2 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 $R^{\mathcal{A}'}_{\mathcal{D}'}(\vec{w})$ for a given $\vec{w}$ by enumerating over the support of $\mathcal{D}'$ as described above. The resulting $R^{\mathcal{A}'}_{\mathcal{D}'}(\vec{w})$ has bit complexity polynomial in the dimension T and the logarithm of $\text{poly}(n, T, 1/\epsilon)$.[9]

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 $\mathcal{A}'$.[10] We argue in the proof of Lemma 17 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 13 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

[9] 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 Chapter 4) makes sure that the latter is non-zero.

[10] Namely, we want that $\mathcal{A}'$ approximately optimizes the linear objective $\vec{w} \cdot \vec{x}$ over $\vec{x} \in F(\mathcal{F}, \mathcal{D}')$ to within a multiplicative factor $\alpha - \gamma$.

complexity $\text{poly}(n, T, \hat{\ell}, \log 1/\epsilon, x)$. Overall,

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

6.8.3 The Runtime of the Revenue-Maximizing LP

Proof of Lemma 17: Ignoring computational efficiency, we can use the construction in Chapter 4 to build a separation oracle for P_0, defined as in Section 6.4. Suppose that we built this separation oracle, SO, and used it to solve the LP of Figure 5-1 with P_0 in place of $F(\mathcal{F}, \mathcal{D}')$ using the ellipsoid algorithm. It follows from [GLS88] 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 $\text{poly}(n, T, \hat{\ell}, \log 1/\epsilon)$.[11] 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 8 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 13 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. □

[11] 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 $\text{poly}(n, T, \hat{\ell}, \log 1/\epsilon)$.

6.9 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 15. *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 16. *The* additive dimension *of an auction setting is the minimum d such that there is an equivalent (by Definition 15) 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 6.1, 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 6.1 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.

6.9.1 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.

6.9.2 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 [BKS12]. 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.

Chapter 7

Conclusion and Open Problems

Motivated by the importance of Mechanism Design for the study of large systems with strategic agents, we investigate one of the its central open problems – how to optimize revenue in multi-dimensional Bayesian mechanisms. After Myerson's seminal work for the single item setting [Mye81], no major progress had been made on this problem for more than 30 years. We generalize Myerson's result to the general case, where there are multiple heterogeneous items for sale. A key contribution of our result is the new framework we have provided for mechanism design by reducing mechanism design problems to algorithm design problems via Linear programming and the ellipsoid method.

We conclude with some open problems raised by the results of this thesis. We begin with the problem motivated by our additive assumption about bidders' valuations.

- What is the broadest class of bidder valuations such that (approximate) revenue-maximization is tractable?

Our result applies to additive valuations, or more generally valuation functions with low additive dimension. It is already a fairly general class, but it does not cover an interesting class – monotone submodular functions. This class has been broadly studied for the welfare-optimal mechanism design problem. In the Bayesian setting, a computational efficient constant factor approximation is known for welfare maximization with only value oracles for monotone submodular functions [HL10, BH11,

HKM11, Von08]. However, in recent work we have shown that revenue maximization is not tractable for all monotone submodular functions. In particular, we show that optimizing revenue with a single monotone submodular bidder is NP-hard to approximate within any polynomial factor [CDW13b]. An interesting future direction is to identify important special classes of monotone submodular functions that still allow efficient revenue maximization. A few potential candidates are budget-additive functions, coverage functions and OXS functions.

Next, we consider a problem motivated by the dependence of our runtime on the number of bidders' types.

- Can we speed up our algorithm when every bidder's values for the items are drawn independently?

In the general case, a bidder's value for different items could be correlated and the natural description for such a distribution is the size of its support. However, when this joint distribution is a product distribution $\times_{j=1}^{n} F_j$, where F_j is the bidder's value distribution for item j, simply describing the F_js is a much more succinct description. In fact, its size could be smaller by an exponential factor.[1] As our algorithm is highly inefficient for these inputs, the revenue-optimal mechanism design problem for this special case remains largely open.

In a recent result, it was shown that an exact solution can not be found in polynomial time, unless $ZPP = P^{\#P}$, even for a very simple case, where there is a single additive bidder whose values for the items are independently distributed on two rational numbers with rational probabilities [DDT13]. However, their hardness result is obtained via a reduction from subset-sum problems, therefore does not exclude even an FPTAS. In fact, PTASes have been found for two related problems: 1) Find the optimal pricing scheme for a single unit demand bidder whose values for the items are MHR[2] independently distributed [CD11]; 2) find the revenue-optimal auction

[1] For example suppose that, for each j, F_j's support size is 2. $O(n)$ numbers suffice to describe $\times_{j=1}^{n} F_j$, but the total support size of the joint distribution is 2^n.

[2] Monotone Hazard Rate (MHR) is a commonly used class of distributions in Economics containing familiar distributions like Gaussian, exponential and uniform. The formal definition is that $\frac{f(x)}{1-F(x)}$ is monotonically decreasing, where $f(\cdot)$ is the density function and $F(\cdot)$ is the cumulative function.

with a constant number of additive bidders whose values are MHR independently distributed [CH13]. Both results heavily rely on a powerful probability theorem, an Extreme-Value theorem showing that the maximum of a group of independent MHR random variables is concentrated [CD11]. We believe such probability tools will be useful for extending our result to a more general case.

So far, we have raised two problems related to revenue maximization. How about other objectives? Motivated by our framework, we want to ask a more general and important problem.

- What other objectives can be efficiently optimized, maybe via a reduction from mechanism design to algorithm design?

The only objective we have considered in this thesis is revenue, but it is not hard to see that our framework applies to a broader class of objectives. In particular, the same reduction will work for any objective function that is concave over the reduced form auctions, e.g. social welfare, any convex combination of social welfare and revenue. This already covers a large family of important objectives, however, some interesting objectives are still missing, for example the makespan on unrelated machines and the max-min fairness. These objectives are sometimes called non-linear in order to distinguish them from social welfare and revenue, which are linear in the allocation and pricing rules.

Non-linear objectives have already been studied in the literature. Indeed, the seminal paper of Nisan and Ronen has already studied minimizing makespan when scheduling jobs to selfish machines, in a non-Bayesian setting [NR99]. Following this work, a lot of AMD research has focused on non-linear objectives in non-Bayesian settings (see, e.g., [CKV07, ADL12] and their references), but positive results have been scarce. Recently, it is shown in [CIL12] that no polynomial-time black-box reduction from truthfully maximizing a non-linear objective to non-truthfully maximizing the same non-linear objective exists without losing a polynomial factor in the approximation ratio, even in Bayesian settings. Even more recently, a non-black box approach was developed in [CHMS13] to approximately minimize makespan in certain Bayesian

settings. [3]

A modified version of our black-box approach sidesteps the hardness result of [CIL12] by reducing the problem of truthfully maximizing an objective to non-truthfully maximizing a modified objective [CDW13b]. Using this reduction, we design an optimal mechanism for fractional max-min fairness. For makespan, our reduction will reduce it to an algorithmic optimization problem. Our new approach also works for makespan. However, the algorithmic problem it reduces it to involves solving an integral program which we do not know how to efficiently approximate. But if given any solution to this algorithmic optimization problem, we can turn it into a solution for the makespan problem with the same approximation ratio. We believe our approach will be useful for obtaining a better solution for the makespan problem, and possibly many other non-linear objectives.

[3]Their mechanism provides a super-constant approximation.

Appendix A

Omitted Details from Chapter 4

A.1 Omitted Details from Section 4.2

Before giving the proof of Theorem 11 we present a useful lemma.

Lemma 19. *Let P be a polytope and $H_1, \ldots, H_i$ be i hyperplanes of the form $\vec{w}_j \cdot \vec{v} = h_j$ such that every point $\vec{x} \in P$ satisfies $\vec{x} \cdot \vec{w}_j \leq h_j$. Then for any $c_1, \ldots, c_i > 0$, any $\vec{a} \in P$ satisfying:*

$$\vec{a} \cdot \left(\sum_{j=1}^{i} c_j \vec{w}_j \right) = \sum_{j=1}^{i} c_j h_j$$

is in $\cap_{j=1}^{i} H_j$.

Proof. Because all $\vec{a} \in P$ satisfy $\vec{a} \cdot \vec{w}_j \leq h_j$ for all j and $c_j > 0 \ \forall j$, the only way to have $\vec{a} \cdot \left(\sum_j c_j \vec{w}_j \right) = \sum_j c_j h_j$ is to have $\vec{a} \cdot c_j \vec{w}_j = c_j h_j$ for all j. Thus, we must have $\vec{a} \cdot \vec{w}_j = h_j$ for all j, meaning that $\vec{a} \in \cap_j H_j$. □

Proof of Theorem 11: Let $\vec{\pi}, \vec{w}'$ denote the output of the corner oracle. First, we observe that if $H_1, \ldots, H_a$ intersect inside $F(\mathcal{F}, \mathcal{D})$, there is some reduced form $\vec{\pi}'$ satisfying $\vec{\pi}' \cdot \vec{w}_j = h_j$ for all j. Therefore, such a reduced form must also satisfy $\vec{\pi}' \cdot \vec{w} = \frac{1}{a} \sum_{j=1}^{a} h_j$. Second, as no feasible reduced form can have $\vec{v} \cdot \vec{w}_j > h_j$, we also get that no feasible reduced form has $\vec{v} \cdot \vec{w} > \frac{1}{a} \sum_{j=1}^{a} h_j$. Putting these two observations together, we see that there exists a $\vec{\pi}'$ with $\vec{\pi}' \cdot \vec{w} = \frac{1}{a} \sum_{j=1}^{a} h_j$, and this is the maximum

over all feasible reduced forms. Therefore, the reduced form $R_{\mathcal{F}}(\vec{w})$ of $VVCG_{\mathcal{F}}(\vec{w})$ necessarily has $R_{\mathcal{F}}(\vec{w}) \cdot \vec{w} = \frac{1}{a} \sum_{j=1}^{a} h_j$. Lemma 4 tells us that $\vec{\pi}$ is the reduced form of a simple virtual VCG allocation rule $VVCG_{\mathcal{F}}(\vec{w}')$, which also maximizes $\vec{x} \cdot \vec{w}$ over all feasible reduced forms $\vec{x}$. Therefore, $\vec{\pi} \cdot \vec{w} = \frac{1}{a} \sum_{j=1}^{a} h_j$ and by Lemma 19, $\vec{\pi}$ is in $\cap_{j=1}^{a} H_j$. From Proposition 6, we know $\vec{\pi}$ is a corner. As each coordinate of $\vec{w}_j$ is a rational number of bit complexity b, and $a \leq n \sum_{i=1}^{m} |T_i|$, we see that each coefficient of $\vec{w}$ is a rational number of bit complexity $\text{poly}(\log(n \sum_{i=1}^{m} |T_i|), b)$. Lemma 4 then guarantees that each coefficient of $\vec{w}'$ is a rational number of bit complexity $\text{poly}(n \sum_{i=1}^{m} |T_i|, b, \ell)$. $\square$

A.2 Proofs Omitted From Section 4.3.1: Exact Implementation

We bound the running time of the algorithms of Section 4.2 when $\mathcal{D}$ is a possibly correlated, uniform distribution. Before doing this, we establish a useful lemma.

Lemma 20. *For all $\mathcal{F}$ and $\mathcal{D}$, if every corner of $F(\mathcal{F}, \mathcal{D})$ is a vector of rational numbers of bit complexity b, the probabilities used by $\mathcal{D}$ have bit complexity ℓ, and SO's input $\vec{\pi}$ is a vector of rational numbers of bit complexity c, then the following are true.*

1. *The separation oracle SO of Section 4.2.1 can be implemented to run in time polynomial in $n\sum_{i=1}^{m}|T_i|$, b, c, ℓ, $|\mathcal{D}|$, and $rt_{\mathcal{F}}\left(poly(n\sum_{i=1}^{m}|T_i|, b, c, \ell)\right)$. Furthermore, the coefficients of any hyperplane that can be possibly output by SO have bit complexity $poly(n\sum_{i=1}^{m}|T_i|, b)$.*

2. *If the corner oracle CO of Section 4.2.2 only takes as input hyperplanes output by SO, it can be implemented to run in time polynomial in $n\sum_{i=1}^{m}|T_i|$, b, ℓ, $|\mathcal{D}|$, and $rt_{\mathcal{F}}\left(poly(n\sum_{i=1}^{m}|T_i|, b, \ell)\right)$.*

Proof. We first bound the runtime of SO, using Theorem 1. The separation oracle is a linear program with $1 + n\sum_{i=1}^{m}|T_i|$ variables, $2n\sum_{i=1}^{m}|T_i|$ constraints, and an internal separation oracle $\widehat{SO}$. $\widehat{SO}$ on input $(\vec{w}, t)$ simply checks if $R_{\mathcal{F}}(\vec{w}') \cdot \vec{w}$, where $\vec{w}'$ is the perturbation of $\vec{w}$ according to Lemma 4, is smaller than or equal to t. If not, it outputs the separation hyperplane $(R_{\mathcal{F}}(\vec{w}'), -1)(\vec{w}, y) \leq 0$. Given that $R_{\mathcal{F}}(\vec{w}')$ is a corner of the polytope and corners have bit complexity b, Theorem 1 tells us that $\widehat{SO}$ will only be called on $\vec{w}, t$ whose coordinates are rational numbers of bit complexity at most $\text{poly}(n\sum_{i=1}^{m}|T_i|, \max\{b, c\})$. To compute $R_{\mathcal{F}}(\vec{w}')$ exactly we can enumerate every profile in the support of $\mathcal{D}$, run $VVCG_{\mathcal{F}}(\vec{w}')$, and see if bidder i was awarded item j, for all i, j. As the coordinates of $\vec{w}'$ are rational numbers of bit complexity $\text{poly}(n\sum_{i=1}^{m}|T_i|, \max\{b, c, \ell\})$ (after Lemma 4 was applied to $\vec{w}$)), this method exactly computes $R_{\mathcal{F}}(\vec{w}')$ in time polynomial in $n\sum_{i=1}^{m}|T_i|, |\mathcal{D}|$, b, c, ℓ

and $rt_{\mathcal{F}}(\text{poly}(n\sum_{i=1}^{m}|T_i|, \max\{b,c,\ell\}))$. After computing $R_{\mathcal{F}}(\vec{w}')$, $\widehat{SO}$ simply takes a dot product and makes a comparison, so the total runtime of SO is polynomial in $n\sum_{i=1}^{m}|T_i|, b, c, \ell, |\mathcal{D}|$ and $rt_{\mathcal{F}}(\text{poly}(n\sum_{i=1}^{m}|T_i|, \max\{b,c,\ell\}))$. Also, by Lemma 5, we know that all hyperplanes output by SO have coefficients that are rational numbers of bit complexity $\text{poly}(n\sum_{i=1}^{m}|T_i|, b)$, which is independent of c.

The corner oracle of Section 4.2.2 has three steps. The first step is simply computing the average of at most $n\sum_{i=1}^{m}|T_i|$ vectors in $\mathbb{R}^{n\sum_{i=1}^{m}|T_i|}$, whose coordinates are rational numbers of bit complexity $\text{poly}(n\sum_{i=1}^{m}|T_i|, b)$ (by the previous paragraph). The second step is applying Lemma 4 to the averaged weight vector to get $\vec{w}'$. So each weight of $\vec{w}'$ is a rational number of bit complexity $\text{poly}(n\sum_{i=1}^{m}|T_i|, \ell, b)$. The last step is computing $R_{\mathcal{F}}(\vec{w}')$. It is clear that the first two steps can be implemented in the desired runtime. As the coordinates of $\vec{w}'$ are rational numbers of bit complexity $\text{poly}(n\sum_{i=1}^{m}|T_i|, b, \ell)$, we can use the same method as in the previous paragraph to compute $R_{\mathcal{F}}(\vec{w}')$ in time polynomial in $n\sum_{i=1}^{m}|T_i|, |\mathcal{D}|$, b, ℓ and $rt_{\mathcal{F}}(\text{poly}(n\sum_{i=1}^{m}|T_i|, b, \ell))$, to implement CO in the desired runtime. □

Corollary 15. *For all $\mathcal{F}$, if $\mathcal{D}$ is a (possibly correlated) uniform distribution over k profiles (possibly with repetitions), and SO's input $\vec{\pi}$ is a vector of rational numbers of bit complexity c, then the following are true.*

1. *The separation oracle SO of Section 4.2.1 can be implemented to run in time polynomial in $n\sum_{i=1}^{m}|T_i|$, k, c and $rt_{\mathcal{F}}(poly(n\sum_{i=1}^{m}|T_i|, \log k, c))$. Furthermore, the coefficients of any hyperplane that can be possibly output by SO have bit complexity $poly(n\sum_{i=1}^{m}|T_i|, \log k)$.*

2. *If the corner oracle CO of Section 4.2.2 only takes as inputs hyperplanes output by SO as input, it can be implemented in time polynomial in $n\sum_{i=1}^{m}|T_i|$, k, and $rt_{\mathcal{F}}(poly(n\sum_{i=1}^{m}|T_i|, \log k))$.*

Proof of Corollary 15: Every corner of $F(\mathcal{F}, \mathcal{D})$ is the reduced form of a deterministic mechanism. So let us bound the bit complexity of the reduced form π of a deterministic mechanism M. We may let $n_{ij}(A)$ denote the number of profiles (with repetition) in the support of $\mathcal{D}$ where bidder i's type is A, and M awards item j to i,

and let $d_{ij}(A)$ denote the number of profiles where bidder i's type is A. Then for all $i, j, A \in T_i$, $\pi_{ij}(A) = \frac{n_{ij}(A)}{d_{ij}(A)}$. As $n_{ij}(A)$ and $d_{ij}(A)$ are integral and at most k, $\pi_{ij}(A)$ has bit complexity $O(\log k)$. So we may take $b = O(\log k)$, $\ell = O(\log k)$, $|\mathcal{D}| = k$, and apply Lemma 20. $\square$

Next we bound the running time of the decomposition algorithm.

Corollary 16. *For all $\mathcal{F}$, if $\mathcal{D}$ is a (possibly correlated) uniform distribution over k profiles (possibly with repetitions), then given a reduced form $\vec{\pi} \in F(\mathcal{F}, \mathcal{D})$, which is a vector of rational numbers with bit complexity c, we can rewrite $\vec{\pi}$ as a convex combination of corners of $F(\mathcal{F}, \mathcal{D})$ using the geometric algorithm of Theorem 2 with running time polynomial in $n \sum_{i=1}^{m} |T_i|$, k, c and $rt_{\mathcal{F}}(poly(n \sum_{i=1}^{m} |T_i|, \log k, c))$.*

Proof of Corollary 16: From Corollary 15 it follows that the coefficients of any hyperplane that can be possibly output by SO have bit complexity $\text{poly}(n \sum_{i=1}^{m} |T_i|, \log k)$. So to apply Theorem 2 it suffices to bound the running time of SO and CO on vectors of rational numbers of bit complexity $c' = \text{poly}(n \sum_{i=1}^{m} |T_i|, \log k, c)$. Using Corollary 15 this is polynomial in $n \sum_{i=1}^{m} |T_i|$, k, c and $rt_{\mathcal{F}}(\text{poly}(n \sum_{i=1}^{m} |T_i|, \log k, c))$. Combining this bound with Theorem 2 finishes the proof. $\square$

Bibliography

[ADL12] Itai Ashlagi, Shahar Dobzinski, and Ron Lavi. Optimal lower bounds for anonymous scheduling mechanisms. *Mathematics of Operations Research*, 37(2):244–258, 2012.

[AFH+12] 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.

[Ala11] 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.

[Arm96] Mark Armstrong. Multiproduct nonlinear pricing. *Econometrica*, 64(1):51–75, January 1996.

[Arm99] Mark Armstrong. Price discrimination by a many-product firm. *Review of Economic Studies*, 66(1):151–68, January 1999.

[Bas01] Suren Basov. Hamiltonian approach to multi-dimensional screening. *Journal of Mathematical Economics*, 36(1):77–94, September 2001.

[BCKW10] Patrick Briest, Shuchi Chawla, Robert Kleinberg, and S. Matthew Weinberg. Pricing Randomized Allocations. In *the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2010.

[BGGM10] 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.

[BH11] Xiaohui Bei and Zhiyi Huang. Bayesian Incentive Compatibility via Fractional Assignments. In *the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2011.

[BKS12] 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.

[BKV05] 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.

[BLP06] Moshe Babaioff, Ron Lavi, and Elan Pavlov. Single-value combinatorial auctions and implementation in undominated strategies. In *the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2006.

[Bor91] Kim C. Border. Implementation of reduced form auctions: A geometric approach. *Econometrica*, 59(4):1175–1187, 1991.

[Bor07] Kim C. Border. Reduced Form Auctions Revisited. *Economic Theory*, 31:167–181, 2007.

[CD11] 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.

[CDW12a] 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.

[CDW12b] 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.

[CDW13a] Yang Cai, Constantinos Daskalakis, and S. Matthew Weinberg. Reducing Revenue to Welfare Maximization : Approximation Algorithms and other Generalizations. In *the 24th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2013.

[CDW13b] Yang Cai, Constantinos Daskalakis, and S. Matthew Weinberg. Understanding Incentives: Mechanism Design becomes Algorithm Design. In *the 54th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2013.

[CH13] Yang Cai and Zhiyi Huang. Simple and Nearly Optimal Multi-Item Auctions. In *the 24th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2013.

[CHK07] Shuchi Chawla, Jason D. Hartline, and Robert D. Kleinberg. Algorithmic Pricing via Virtual Valuations. In *the 8th ACM Conference on Electronic Commerce (EC)*, 2007.

[CHMS10] 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.

[CHMS13] Shuchi Chawla, Jason Hartline, David Malec, and Balasubramanian Sivan. Prior-Independent Mechanisms for Scheduling. In *Proceedings of 45th ACM Symposium on Theory of Computing (STOC)*, 2013.

[CIL12] Shuchi Chawla, Nicole Immorlica, and Brendan Lucier. On the limits of black-box reductions in mechanism design. In *Proceedings of the 44th Symposium on Theory of Computing (STOC)*, 2012.

[CKM11] Yeon-Koo Che, Jinwoo Kim, and Konrad Mierendorff. Generalized Reduced-Form Auctions: A Network-Flow Approach. *University of Zürich, ECON-Working Papers*, 2011.

[CKV07] George Christodoulou, Elias Koutsoupias, and Angelina Vidali. A lower bound for scheduling mechanisms. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1163–1170. Society for Industrial and Applied Mathematics, 2007.

[Cla71] Edward H. Clarke. Multipart pricing of public goods. *Public Choice*, 11(1):17–33, 1971.

[CM85] Jacques Cremer and Richard P. McLean. Optimal selling strategies under uncertainty for a discriminating monopolist when demands are interdependent. *Econometrica*, 53(2):345–361, 1985.

[CM88] Jacques Cremer and Richard P. McLean. Full extraction of the surplus in bayesian and dominant strategy auctions. *Econometrica*, 56(6):1247–1257, 1988.

[CMS10] Shuchi Chawla, David L. Malec, and Balasubramanian Sivan. The Power of Randomness in Bayesian Optimal Mechanism Design. In *the 11th ACM Conference on Electronic Commerce (EC)*, 2010.

[DD09] Shahar Dobzinski and Shaddin Dughmi. On the power of randomization in algorithmic mechanism design. In *FOCS*, pages 505–514, 2009.

[DDT13] Constantinos Daskalakis, Alan Deckelbaum, and Christos Tzamos. The Complexity of Optimal Mechanism Design. *Manuscript: http://arxiv.org/pdf/1211.1703v2.pdf*, 2013.

[DN07] Shahar Dobzinski and Noam Nisan. Mechanisms for multi-unit auctions. In *ACM Conference on Electronic Commerce*, pages 346–351, 2007.

[DNS05] Shahar Dobzinski, Noam Nisan, and Michael Schapira. Approximation algorithms for combinatorial auctions with complement-free bidders. In *STOC*, pages 610–618, 2005.

[DNS06] Shahar Dobzinski, Noam Nisan, and Michael Schapira. Truthful randomized mechanisms for combinatorial auctions. In *STOC*, pages 644–652, 2006.

[DR10] Shaddin Dughmi and Tim Roughgarden. Black-box randomized reductions in algorithmic mechanism design. In *51st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2010.

[DW12] Constantinos Daskalakis and S. Matthew Weinberg. Symmetries and Optimal Multi-Dimensional Mechanism Design. In *the 13th ACM Conference on Electronic Commerce (EC)*, 2012.

[GLS81] Martin Grötschel, László Lovász, and Alexander Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1(2):169–197, 1981.

[GLS88] Martin Grötschel, Lászlo Lovász, and Alexander Schrijver. *Geometric Algorithms and Combinatorial Optimization*, volume 2 of *Algorithms and Combinatorics*. Springer, 1988.

[Gro73] Theodore Groves. Incentives in teams. *Econometrica*, 41(4):617–631, 1973.

[HKM11] Jason D. Hartline, Robert Kleinberg, and Azarakhsh Malekian. Bayesian Incentive Compatibility via Matchings. In *the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2011.

[HL10] Jason D. Hartline and Brendan Lucier. Bayesian Algorithmic Mechanism Design. In *the 42nd ACM Symposium on Theory of Computing (STOC)*, 2010.

[HN12] Sergiu Hart and Noam Nisan. Approximate Revenue Maximization with Multiple Items. In *the 13th ACM Conference on Electronic Commerce (EC)*, 2012.

[Hoe63] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

[Jan02] 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.

[Kaz01] Eiichiro Kazumori. Optimal auction for heterogeneous objects. Working papers, 2001.

[KW12] Robert Kleinberg and S. Matthew Weinberg. Matroid Prophet Inequalities. In *the 44th Annual ACM Symposium on Theory of Computing (STOC)*, 2012.

[LOS02] Daniel J. Lehmann, Liadan O'Callaghan, and Yoav Shoham. Truth revelation in approximately efficient combinatorial auctions. *J. ACM*, 49(5):577–602, 2002.

[LS05] Ron Lavi and Chaitanya Swamy. Truthful and near-optimal mechanism design via linear programming. In *FOCS*, pages 595–604, 2005.

[Mat84] Steven Matthews. On the Implementability of Reduced Form Auctions. *Econometrica*, 52(6):1519–1522, 1984.

[MM88] R. Preston McAfee and John McMillan. Multidimensional incentive compatibility and mechanism design. *Journal of Economic Theory*, 46(2):335–354, December 1988.

[MR84] Eric Maskin and John Riley. Optimal Auctions with Risk Averse Buyers. *Econometrica*, 52(6):1473–1518, 1984.

[MR92] R. Preston McAfee and Philip J. Reny. Correlated information and mechanism design. *Econometrica*, 60(2):395–421, 1992.

[MV06] Alejandro M. Manelli and Daniel R. Vincent. Bundling as an optimal selling mechanism for a multiple-good monopolist. *Journal of Economic Theory*, 127(1):1–35, March 2006.

[MV07] 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.

[Mye79] Roger B Myerson. Incentive compatibility and the bargaining problem. *Econometrica*, 47(1):61–73, January 1979.

[Mye81] Roger B. Myerson. Optimal Auction Design. *Mathematics of Operations Research*, 6(1):58–73, 1981.

[NR99] Noam Nisan and Amir Ronen. Algorithmic Mechanism Design (Extended Abstract). In *Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing (STOC)*, 1999.

[RC98] Jean-Charles Rochet and Philippe Chone. Ironing, sweeping, and multidimensional screening. *Econometrica*, 66(4):783–826, July 1998.

[Sto] Brad Stone. Pakistan cuts access to youtube worldwide. *New York Times*.

[Tha04] John Thanassoulis. Haggling over substitutes. *Journal of Economic Theory*, 117(2):217–245, August 2004.

[Vic61] William Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *The Journal of Finance*, 16(1):8–37, 1961.

[Von08] Jan Vondrák. Optimal approximation for the submodular welfare problem in the value oracle model. In *STOC*, pages 67–74, 2008.

[Wil93] Robert Wilson. *Nonlinear Pricing*. New York: Oxford University Press, 1993.

[Zhe00] Charles Z. Zheng. Optimal auction in a multidimensional world. Working papers, January 2000.