Algorithms for Strategic Agents

by

S. Matthew Weinberg

B.A., Cornell Unversity (2010)

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 2014

Author . . . .
Department of Electrical Engineering and Computer Science
May 21, 2014

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

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

Algorithms for Strategic Agents

by

S. Matthew Weinberg

Submitted to the Department of Electrical Engineering and Computer Science
on May 21, 2014, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Computer Science

Abstract

In traditional algorithm design, no incentives come into play: the input is given, and your algorithm must produce a correct output. How much harder is it to solve the same problem when the input is not given directly, but instead reported by strategic agents with interests of their own? The unique challenge stems from the fact that the agents may choose to lie about the input in order to manipulate the behavior of the algorithm for their own interests, and tools from *Game Theory* are therefore required in order to predict how these agents will behave.

We develop a new algorithmic framework with which to study such problems. Specifically, we provide a computationally efficient black-box reduction from solving any optimization problem on "strategic input," often called *algorithmic mechanism design* to solving a perturbed version of that same optimization problem when the input is directly given, traditionally called *algorithm design*.

We further demonstrate the power of our framework by making significant progress on several long-standing open problems. First, we extend Myerson's celebrated characterization of single item auctions [Mye81] to multiple items, providing also a computationally efficient implementation of optimal auctions. Next, we design a computationally efficient 2-approximate mechanism for job scheduling on unrelated machines, the original problem studied in Nisan and Ronen's paper introducing the field of Algorithmic Mechanism Design [NR99]. This matches the guarantee of the best known computationally efficient algorithm when the input is directly given. Finally, we provide the first hardness of approximation result for optimal mechanism design.

Thesis Supervisor: Constantinos Daskalakis
Title: Associate Professor of EECS

Acknowledgments

To start, I would like to thank my advisor, Costis Daskalakis, for his belief in me and all his guidance in pushing me to be excellent. When I first arrived at MIT, Costis and I discussed research directions. At the time, I was confident that I possessed the tools to make minor improvements on prior work and excitedly told him this. Chuckling politely, he told me that this was great, but that we should strive to solve more important problems. We worked long and hard that year, often going long stretches without any progress, but by the end we had started down the path that would eventually culminate in this thesis. Thank you, Costis, for teaching me to aim high in my pursuits, to always push the boundaries of my comfort zone and for guiding me along this journey.

Special thanks are also due to Bobby Kleinberg, who introduced me to theoretical computer science and taught me how to do research during my time at Cornell. His mix of patience and enthusiasm turned me from an overexcited problem-solver into a competent researcher. Today, Bobby is still one of my most trusted mentors and I am truly grateful for everything that he has done for me. Thank you as well to everyone else who has mentored me through this journey: Moshe Babaioff, Jason Hartline, Nicole Immorlica, Brendan Lucier, Silvio Micali, Christos Papadimitriou and Éva Tardos; especially to Parag Pathak and Silvio for serving on my dissertation committee, and Nicole and Brendan for an amazing summer internship at Microsoft Research.

I wish to thank the entire theory group at MIT. Thank you in particular to Costis, Silvio, Shafi Goldwasser, Piotr Indyk, Ankur Moitra, and Madhu Sudan for always being available to provide advice. Thanks as well to everyone who I've had the pleasure of collaborating with: Moshe, Costis, Nicole, Brendan, Bobby, Silvio, Pablo Azar, Patrick Briest, Yang Cai, Shuchi Chawla, Michal Feldman, and Christos Tzamos. A special thanks to Yang, without whom I'm certain the work in this thesis would not have been possible. I also thank him for teaching me everything there is to know about EECS at MIT - without him, I would be lost.

Thanks as well to the students of MIT, Cornell, and elsewhere who made work so enjoyable: Pablo Azar, Alan Deckelbaum, Michael Forbes, Hu Fu, Nima Haghpanah, Darrell Hoy, G Kamath, Yin-Tat Lee, Matthew Paff, Pedram Razavi, Ran Shorrer, Aaron Sidford, Greg Stoddard, Vasilis Syrgkanis, Sam Taggart, and Di Wang.

Thank you to all my friends from Pikesville, Cornell, and MIT for supporting me through the difficulties of research: Alex, Brandon, Josh, Megan, Shen, Andy, Ganesh, Farrah, Henry, Jason, Job, Steve, Meelap, and Tahin. Thank you to MIT and Cornell Sport Taekwondo for teaching me to pursue excellence in all aspects of life. Thanks especially to my coaches, Master Dan Chuang and Master Han Cho.

I must thank Jennie especially for her unending love and support over the past five years. Her presence in my life made it possible to weather the unavoidable hardships that accompany research with determination and hope for the future.

Finally, words cannot express how grateful I am to my family for their incredible support over this long journey. I am eternally grateful to Mom, Dad, Grandma, Alec, Syrus, Dagmara and Myles for everything that they've done to support me; for their love, care, and encouragement. I wish to dedicate this thesis to them.

Contents

List of Figures

Chapter 1

Introduction

Traditionally, *Algorithm Design* is the task of producing a correct output from a given input in a computationally efficient manner. For instance, maybe the input is a list of integers and the desired output is the maximum element in that list. Then a good algorithm, findMax, simply scans the entire list and outputs the maximum. A subtle, but important assumption made by the findMax algorithm is that the input is accurate; that is, the input list is exactly the one for which the user wants to find the maximum element.

While seemingly innocuous, this assumption implicitly requires the algorithm to be completely oblivious both to where the input comes from and to what the output is used for. Consider for example a local government with a stimulus package that they want to give to the company that values it most. They could certainly try soliciting from each company their value for the package, making a list, running the findMax algorithm, and selecting the corresponding company. After all, findMax will correctly find the company with the largest reported value. But something is obviously flawed with this approach: why would a company report accurately their value when a larger report will make them more likely to win the package? Basic game theory tells us that each company, if strategic, will report a bid of ∞, and the winner will just be an arbitrary company.

The issue at hand here is *not* the validity of findMax: the maximum element of $\{\infty, \dots, \infty\}$ is indeed ∞. Instead, the problem is that separating the design and application of findMax is a mistake. The companies control the input to findMax, the input to findMax affects the winner of the stimulus package, and the companies care who wins the package. So it's in each company's interest to manipulate the algorithm by misreporting their input.

A well-cited example of this phenomenon occured in 2008 when a strategic table update in Border Gateway Protocol (BGP), the protocol used to propagate routing information throughout the Internet, caused a global outage of YouTube [McC08, McM08]. What went wrong? At the time, offensive material prompted Pakistani Internet Service Providers to censor YouTube within Pakistan, updating their BGP tables to map from www.youtube.com to nowhere insted of correctly to YouTube. The outage occurred when this update inadvertantly spread to the rest of the Internet as well. What's the point? BGP works great when everyone correctly updates their tables, so there are no problems purely from the algorithmic perspective. But when users instead updated their tables to serve their own interests, the protocol failed.

A more recent example occured in 2013, when an entire university class of students boycotted an Intermediate Programming final in order to manipulate the professor's grading policy [Bud13]. What went wrong? The professor's policy curved the highest grade to 100 and bumped up all others by the same amount. The students' boycott guaranteed that their 0s would all be curved to 100s. What's the point? Algorithmically, grading on a curve is a great way to overcome inconsistencies across different course offerings, so the system works well when all students perform honestly. But when the students performed strategically, the resulting grades were meaningless.

Should *every* algorithm be implemented in a way that is robust to strategic manipulation? Of course not. Sometimes data is just data and the input is directly given. But as the Internet continues to grow, more and more of our data comes from users with a vested interest in the outcome of our algorithms. Motivated by examples like those above, we are compelled to design algorithms robust to potential manipulation by strategic agents in these settings. We refer to such algorithms as *mechanisms*, and address in this thesis the following fundamental question:

Question 1. *How much more difficult is mechanism design than algorithm design?*

1.1 Mechanism Design

Before attempting to design mechanisms robust against strategic behavior, we must first understand how strategic agents in fact behave. This daunting task is the focus of *Game Theory*, which aims to develop tools for predicting how strategic agents will interact within a given environment or system. *Mechanism Design*, sometimes referred to as "reverse game theory," instead aims to design systems where agents' behavior is easily predictable, and this behavior results in a desireable outcome. Returning to our stimulus package example, we see that findMax by itself is not a good mechanism: even though each company will predictably report a value of ∞, the outcome is highly undesireable.

Still, a seminal paper of Vickrey [Vic61] provides a modification of the findMax algorithm that awards the package to the company with the highest value *even in the face of strategic play*. Vickrey showed that if in addition to awarding the package to the highest bidder, that same bidder is also charged a price equal to the second-largest value reported, then the resulting mechanism, called the second-price auction, is *truthful*. That is, it is in each company's best interest to report their true value. Therefore, even strategic agents will tell the truth, and the package will always go to the company with the highest value.

Remarkably, Vickrey's result was extended by Clarke [Cla71] and Groves [Gro73] far beyond single-item settings, generalizing the second-price auction to what is now called the Vickrey-Clarke-Groves (VCG) mechanism. Imagine a designer choosing from any set $\mathcal{F}$ of possible outcomes, and that each agent i has a private value $t_i(x)$ should the outcome x be selected. Then if the designer's goal is to select the outcome maximizing the total *welfare*, which is the sum of all agents' values for the outcome selected and can be written as $\sum_i t_i(x)$, the VCG mechanism is again truthful and always selects the optimal outcome. The mechanism simply asks each agent to report their value $t_i(x)$ for each outcome and selects the outcome $\operatorname{argmax}_{x\in\mathcal{F}}\{\sum_i t_i(x)\}$. It turns out that this selection procedure can be complemented by a pricing scheme to make a truthful mechanism; one example of such a pricing scheme is called the Clarke pivot rule [Cla71]. Interestingly, one can view the VCG mechanism as a *reduction* from mechanism to algorithm design: with black-box access[1] to an optimal *algorithm* for welfare, one can execute, computationally efficiently, an optimal *mechanism* for welfare. That is, given black-box access to an algorithm that optimizes welfare when each $t_i(x)$ is directly given, one can design a computationally efficient mechanism that optimizes welfare when all $t_i(x)$ are instead reported by strategic agents.

1.2 Multi-Dimensional Mechanism Design

What if instead of optimizing the value of the agents, the designer aims to maximize his own revenue? Finding the optimal mechanism in this case is of obvious importance, as millions of users partake in such interactions every day. Before tackling this important problem, we must first address what it means for a mechanism to be optimal in this setting. One natural definition might be that the optimal mechanism obtains revenue equal to the best possible welfare, with each agent paying their value for the outcome selected. However, even in the simplest case of a single agent interested in a single item, this benchmark is unattainable: how can the designer possibly both encourage the agent to report her true value for the item, yet also charge her a price equal to that value?

The underlying issue in attaining this benchmark is that it is *prior-free*. Without any beliefs whatsoever on the agent's value, how can the designer know whether to interact on the scale of pennies, dollars or billions of dollars? And is it even realistic to model the designer as lacking this knowledge? Addressing both issues simultaneously, Economists typically model the designer as having a *Bayesian prior* (i.e. a distribution) over possible values of the agents, and mechanisms are judged based on their expected revenue. The goal then becomes to find the mechanism whose expected revenue is maximized (over all mechanisms).

Within this model, a seminal paper of Myerson [Mye81] provides a strong structural characterization of the revenue-optimal mechanism for selling a single item, showing that it takes the following simple form: first, agents are asked to report their value for the item, then each agent's reported value is transformed via a closed formula to what is called an *ironed virtual value*,[2] and

[1]Black-box access to an algorithm means that one can probe the algorithm with input and see what output it selects, but not see the inner workings of the algorithm.

[2]The precise transformation mapping an agent's reported value to its corresponding ironed virtual value depends

finally the item is awarded to the agent with the highest non-negative ironed virtual value (if any). Furthermore, in the special case that the designer's prior is independent and identically distributed across agents and the marginals satisfy a technical condition called regularity,[3] the optimal auction is exactly a second price auction with a reserve price. That is, the item is awarded to the highest bidder if he beats the reserve, and he pays the maximum of the second highest bid and the reserve. If no agent beats the reserve, the item is unsold. Interestingly, one can also view Myerson's optimal auction as a reduction from mechanism to algorithm design: with black-box access to the findMax algorithm, one can execute the optimal mechanism for revenue computationally efficiently.

While Myerson's result extends to what are called *single-dimensional* settings (that include for example the sale of k identical items instead of just one), it does not apply even to the sale of just two heterogeneous items to a single agent. Thirty-three years later, we are still quite far from a complete understanding of optimal multi-item mechanisms. Unsurprisingly this problem, dubbed the *optimal multi-dimensional mechanism design problem* is considered one of the central open problems in Mathematical Economics today.

Question 2. *(Multi-Dimensional Mechanism Design) What is the revenue-optimal auction for selling multiple items?*

This thesis makes a substantial contribution towards answering this question. We generalize Myerson's characterization to multi-item settings, albeit with some added complexity, and show how to execute the optimal auction computationally efficiently. We provide in Section 1.4 further details regarding our contributions, taking a brief detour in the following section to acclimate the reader with some surprising challenges in multi-dimensional mechanism design.

1.2.1 Challenges in Multi-Dimensional Mechanism Design

Already in 1981, Myerson's seminal work provided the revenue-optimal auction for many bidders in single-dimensional settings. Yet thirty-three years later, a complete understanding of how to sell just two items to a single buyer remains elusive. That is not to say that progress has not been made

on the Bayesian prior on the value of that agent. The precise formula for this transformation is not used in this thesis.

[3] A one-dimensional distribution with CDF F and PDF f is regular if $v - \frac{1-F(v)}{f(v)}$ is monotonically non-decreasing.

(see Section 2.4 in Chapter 2 for a summary of this progress), but prior to this thesis we were still quite far from any understanding in unrestricted settings. The goal of this section is to overview some difficulties in transitioning from single- to multi-dimensional mechanism design.

We do so by presenting several examples involving *additive* agents. An agent if additive if they have some (private) value v_i for receiving each item i, and their value for receiving a set S of items is $\sum_{i \in S} v_i$. Recall that in principle the agent's private information could be some arbitrary function mapping each set of items to the value that agent has for that set. So additive agents are a special (but also very important) case. Drawing an analogy to welfare maximization, one might hope that settings with additive agents are considerably better behaved than more general settings. Indeed, if our goal was to maximize welfare, the VCG mechanism would simply award each item to the agent with the highest value (for that item) and charge her the second highest bid (for that item). In other words, the welfare optimal auction for many items and additive agents simply runs several independent instances of the Vickrey auction. So does the revenue optimal auction for many items and additive agents simply run several independent instances of Myerson's auction? Unfortunately, even in the case of a single additive agent and two items, the optimal auction can be *significantly* more complex.

Lack of Indepencence. Consider a seller with two items for sale to a single additive buyer. Assume further that the distribution of v_1, the buyer's value for item 1, is independent of v_2, the buyer's value for item 2. Then as far as the buyer is concerned, there is absolutely no interaction between the items: her value for receiving item 1 is completely independent of whether or not she receives item 2. And furthermore her value for item 2 yields absolutely no information about her value for item 1. It's natural then to think that there should be no interaction between the items in the optimal mechanism. However, this is false, as the following simple example shows.

Example: There is a single additive buyer whose value for each of two items is drawn independently from the uniform distribution on $\{1, 2\}$. Then the optimal mechanism that treats both items separately achieves expected revenue exactly 2: the seller can sell each item at a price of 1, and have it sell with probability 1, or at a price of 2, and have it sell with probability $1/2$. In either case, the expected per-item revenue is 1. Yet, the mechanism that offers a take-it-or-leave-it price of 3 for both items together sells with probability $3/4$, yielding expected revenue $9/4 > 2$.

It's hard to imagine a simpler multi-dimensional example than that above, and already we see that the optimal mechanism may have counterintuitive properties. One could then ask if there is at least a constant upper bound on the ratio between the optimal revenue and that of the mechanism that optimally sells items separately.

Example: There is a single additive buyer whose value for each of m items is drawn independently from the *equal revenue* distribution, which has CDF $F(x) = 1 - 1/x$ for all $x \geq 1$. Then the optimal mechanism that treats each item separately achieves expected revenue exactly m: the seller can set any price p_i on item i, and have it sell with probability $1/p_i$, yielding an expected per-item revenue of 1. Hart and Nisan [HN12] show that the mechanism offering a take-it-or-leave-it price of $\Theta(m \log m)$ for all m items together sells with probability $1/2$, yielding expected revenue $\Theta(m \log m)$.[4] As $m \to \infty$, the ratio between the two revenues is unbounded.

The takeaway message from these two examples is that just because the input distributions have an extremely simple form *does not* mean that the optimal mechanism shares in any of this simplicity. Specifically, even when there is absolutely no interaction between different items from the perspective of the input, the optimal mechanism may require such interaction anyway.

Non-Monotonicity of Revenue. Compare two instances of the problem where a single agent is interested in a single item: one where her value is drawn from a distribution with CDF F_1, and another where it is drawn from a distribution with CDF F_2. Further assume that F_1 stochastically dominates F_2 (that is, $F_1(x) \leq F_2(x)$ for all x). Which instance should yield higher expected revenue? As F_1 stochastically dominates F_2, one way to sample a value from F_1 is to first sample a value from F_2, and then increase it (by how much depends on the specifics of F_1 and F_2, but such a procedure is always possible). Surely, increasing the agent's value in this manner can't possibly decrease the optimal revenue, right? Right. The optimal revenue of selling a single item to an agent whose value is drawn from F_1 is at least as large as selling instead to an agent whose value is drawn from F_2. Indeed, a corollary of Myerson's characterization is that the optimal mechanism

[4]Throughout this thesis we use the standard notation $f(n) = O(g(n))$ if there exists a universal constant $c > 0$ such that $f(n) \leq c \cdot g(n)$ for all n, and $f(n) = \Omega(g(n))$ if there exists a universal constant $c > 0$ such that $f(n) \geq c \cdot g(n)$. We also write $f(n) = \Theta(g(n))$ if $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$.

for selling a single item to a single agent is to set a take-it-or-leave-it price. If p is the optimal price for F_2, then setting the same price for F_1 yields at least as much revenue, as by definition we have $F_1(p) \leq F_2(p)$.

What if there are two items, and the agent's value for each item is sampled i.i.d. from F_1 vs. F_2? It's still the case that values from $F_1 \times F_1$ can be obtained by first sampling values from $F_2 \times F_2$ and increasing them (by possibly different amounts, again depending on the specifics of F_1 and F_2). And again it seems clear that increasing the agent's values in this way can't possibly decrease the optimal revenue. Mysteriously, this is no longer the case. Hart and Reny [HR12] provide a counter-example: two one-dimensional distributions F_1 and F_2 such that F_1 stochastically dominates F_2, but the optimal revenue is larger for values sampled from $F_2 \times F_2$ than $F_1 \times F_1$.

The takeaway from this discussion is that the optimal revenue may change erratically as a function of the input distribution. Specifically, changes to the input that should "obviously" increase revenue, such as increasing values in a stochastically dominating way, can in fact cause the revenue to decrease.

Role of Randomization. In principle, the designer could try making use of randomization to increase revenue. For instance, maybe instead of simply offering an item at price 100, he could also offer a lottery ticket that awards the item with probability $1/2$ for 25, letting the agent choose which option to purchase. Myerson's characterization implies that for sellling a single item (even to multiple agents), randomization doesn't help: the designer can make just as much revenue by setting a single fixed price. But in the case of multiple items, it does.

Example ([DDT13]). Consider a single additive agent interested in two items. Her value for item 1 is drawn uniformly from $\{1, 2\}$ and her value for item 2 is drawn independently and uniformly from $\{1, 3\}$. Then the unique optimal mechanism allows the agent to choose from the following three options: receive both items and pay 4, or receive the first item with probability 1, the second with probability $1/2$ and pay 2.5, or receive nothing and pay nothing.

Daskalakis, Deckelbaum, and Tzamos further provide an example with a single additive agent and two items where the unique optimal mechanism offers a menu of *infinitely many* lottery tickets for the agent to choose from [DDT13]. And even worse, Hart and Nisan provide an instance with a single additive agent, whose value for each of two items is correlated, where the revenue of the optimal deterministic mechanism is finite, yet the revenue of the optimal randomized mechanism is infinite [HN13].

The takeaway from this discussion is that randomization is not necessary in single-dimensional settings, but quite necessary in multi-dimensional settings. Not only is the unique optimal mechanism randomized in extremely simple examples, but there is a potentially infinite gap between the revenue of the optimal randomized mechanism and that of the optimal deterministic one.

Difficulties like the above three certainly help explain why multi-dimensional mechanism design is so much more challenging than its single-dimensional counterpart, but they also provide evidence that a characterization as strong as Myerson's is unrealistic. In other words, any general characterization of optimal multi-dimensional mechanisms must be rich enough to accommodate all three complications discussed above, and therefore cannot be quite so simple.

1.3 Algorithmic Mechanism Design

The study of welfare and revenue is of clear importance, but the goal of this thesis is to provide a generic connection between mechanism and algorithm design, not limited to any specific objective. In traditional algorithm design (of the style considered in this thesis), one can model the input as having k components, $t_1, \dots, t_k$, with the designer able to choose any outcome $x \in \mathcal{F}$. Based on the input, $\vec{t}$, each outcome x will have some quality $O(\vec{t}, x)$, and the designer's goal is to find a computationally efficient algorithm mapping inputs $\vec{t}$ to outputs in $\text{argmax}_{x \in \mathcal{F}}\{O(\vec{t}, x)\}$. As an example objective, welfare can be written as $O(\vec{t}, x) = \sum_i t_i(x)$.

In *Algorithmic Mechanism Design*, one instead models each component of the input as being private information to a different self-interested agent, and allows the designer to charge prices as well. The designer's goal is still to find a computationally efficient mechanism that optimizes O. Additionally, O may now also depend on the prices charged, so we write $O(\vec{t}, x, \vec{p})$. As an example, revenue can be written as $\sum_i p_i$. Two natural refinements of our motivating Question 1 arise and lie at the heart of Algorithmic Mechanism Design:

Question 3. *How much worse is the quality of solution output by the optimal mechanism versus that of the optimal algorithm?*

Question 4. *How much more (computationally) difficult is executing the optimal mechanism versus the optimal algorithm?*

In this context, one can view the VCG mechanism as providing an answer to both Questions 3 and 4 of "not at all" when the objective is welfare, even in prior-free settings. Indeed, the VCG mechanism guarantees that the welfare-optimal outcome is chosen. And furthermore, the only algorithmic bottleneck in running the VCG mechanism is finding the welfare-optimal outcome for the given input. For revenue, the answer to Question 3 is "infinitely worse," even in Bayesian settings: consider a single buyer whose value for a single item is sampled from the equal revenue curve ($F(x) = 1 - 1/x$). Then the optimal mechanism obtains an expected revenue of 1 no matter what price is set, whereas the optimal algorithm can simply charge the agent her value, obtaining an expected revenue of ∞. Still, Myerson's characterization still shows that the answer to Question 4 is "not at all" in single-dimensional Bayesian settings. In this context, our results extending Myerson's characterization to multi-dimensional mechanism design further provide an answer of "not at all" to Question 4 for revenue in all Bayesian settings as well.

Still, there are many important objectives beyond welfare and revenue that demand study. Nisan and Ronen, in their seminal paper introducing Algorithmic Mechanism Design, study the objective of makespan minimization [NR99]. Here, the designer has a set of m jobs that he wishes to process on k machines in a way that minimizes the makespan of the schedule, which is the time until all jobs have finished processing (or the maximum load on any machine). Already this is a challenging algorithmic problem that received much attention since at least the 1960s [Gra66, Gra69, GJ75, HS76, Sah76, GIS77, GJ78, GLLK79, DJ81, Pot85, HS87, LST87, HS88, ST93a]. The challenge becomes even greater when the processing time of each job on each machine is private information known only to that machine. In sharp contrast to welfare maximization, Nisan and Ronen show that the optimal truthful mechanism can indeed perform worse than the optimal algorithm, conjecturing that it is in fact much worse. A long body of work followed in attempt to prove their conjecture [CKV07, KV07, MS07, CKK07, LY08a, LY08b, Lu09, ADL12], which still remains open today, along with the equally pressing question of whether or not one can execute the optimal mechanism computationally efficiently. Following their work, resolving Questions 3 and 4 has been a central open problem in Algorithmic Mechanism Design.

Our main result addressing these questions is a computationally efficient black-box reduction from mechanism to algorithm design. That is, for any optimization problem, if one's goal is to design a truthful mechanism, one need only design an algorithm for (a perturbed version of) that same optimization problem. While this reduction alone does not provide a complete answer to Questions 3 or 4, it does provide an algorithmic framework with which to tackle them that has already proved incedibly useful. Making use of this framework, we are able to resolve Question 4 for makespan minimization and other related problems with a resounding "not at all."

1.4 Overview of Results

Our most general result is a computationally efficient black-box reduction from mechanism to algorithm design for general optimization problems. But let us begin with an application of our framework to the important problem of multi-dimensional mechanism design. Here, we show the following result, generalizing Myerson's celebrated result to multiple items. In this context, one should interpret the term "virtual welfare" *not* as being welfare computed with respect to the specific virtual values transformed according to Myerson's formula, but instead as simply welfare computed with respect to *some* virtual value functions, which may or may not be the same as the agents' original value functions.

Informal Theorem 1. *The revenue optimal auction for selling multiple items is a distribution over virtual welfare maximizers. Specifically, the optimal auction asks agents to report their valuation functions, transforms these reports into virtual valuations (via agent-specific randomized transformations), selects the outcome that maximizes virtual welfare, and charges prices to ensure that the entire procedure is truthful. Furthermore, when agents are additive, the optimal auction (i.e. the specific transformation to use and what prices to charge based on the designer's prior) can be found computationally efficiently.*

In comparison to Myerson's optimal single-dimensional auction, we show that the optimal multi-dimensional auction has an identical structure, in that its allocation rule is still a virtual-welfare maximizer. The important difference is that in single-dimensional settings the virtual transformations are deterministic and computed according to a closed formula. In multi-dimensional settings, the transformations are randomized, and we show that they can be computed computationally efficiently. Note that the use of randomization cannot be avoided even in extremely simple multi-dimensional settings given the examples of Section 1.2.1. While certainly not as compelling as Myerson's original characterization, Informal Theorem 1 constitutes immense progress on both the structural front (namely, understanding the structure of revenue-optimal auctions) and algorithmic front (namely, the ability to find revenue-optimal auctions efficiently, regardless of their structure). Section 2.4 in Chapter 2 overviews the vast prior work on this problem.

From here, we turn to more general objectives, aiming to describe our complete reduction from mechanism to algorithm design. As we have already discussed, such a reduction is already achieved for welfare by the VCG mechanism [Vic61, Cla71, Gro73]. However the resulting reduction is fragile with respect to approximation: if an approximately optimal welfare-maximizing algorithm is used inside the VCG auction, the resulting mechanism is not a truthful mechanism at all! As it is often NP-hard to maximize welfare in combinatorially rich settings, an *approximation-preserving*[5] reduction would be highly desireable.

Question 5. *Is there an approximation-preserving black-box reduction from mechanism to algorithm design, at least for the special case of welfare?*

Interestingly, the answer to Question 5 depends on the existence of a prior. A recent series of works [PSS08, BDF$^+$10, Dob11, DV12] shows that in prior-free settings, the answer is no. In contrast, a parallel series of works [HL10, HKM11, BH11] shows that in Bayesian settings, the answer is yes. The takeaway message here is two-fold. First, accommodating approximation algorithms in reductions from mechanism to algorithm design is quite challenging even when an exact reduction is already known, as in the case of welfare optimization. And second, to have

[5]In this context, a reduction is approximation preserving if given black-box access to an α-approximation algorithm, the resulting mechanism is truthful, and also an α-approximation.

any hope of accommodating approximation, we must be in a Bayesian setting. Inspired by this, a natural question to ask is the following:

Question 6. *Is there an approximation-preserving black-box reduction from mechanism design for any objective to algorithm design for that same objective in Bayesian settings?*

Unfortunately, recent work shows that the answer to Question 6 is actually no [CIL12]. Specifically, no such reduction can possibly exist for makespan minimization. Have we then reached the limits of black-box reductions in mechanism design, unable to tackle objectives beyond welfare? Informal Theorem 2 below states this is not the case, as we circumvent the impossibility result of [CIL12] by *perturbing the algorithmic objective*.

Informal Theorem 2. *In Bayesian settings, for every objective function O there is a computationally efficient, approximation-preserving black-box reduction from mechanism design optimizing O to algorithm design optimizing O+virtual welfare.*

Informal Theorem 2 provides an algorithmic framework with which to design truthful mechanisms. This thesis continues by making use of this framework to design truthful mechanisms for two paradigmatic algorithm design problems. The first problem we study is that of *job scheduling on unrelated machines*, a specific instance of makespan minimization (from Section 1.3) and the same problem studied in Nisan and Ronen's seminal paper [NR99]. The term "unrelated" refers to the fact that the processing time of job j on machine i is unrelated to the processing time of job j on machine i' or job j' on machine i. Section 4.3 in Chapter 4 overviews the wealth of prior work in designing mechanisms for makespan minimization. We only note here that the best known computationally efficient *algorithm* obtains a 2-approximation.

Informal Theorem 3. *There is a computationally efficient truthful mechanism for job scheduling on unrelated machines that obtains a 2-approximation.*

The second algorithmic problem we study is that of *fair allocation of indivisible goods*. Here, a set of m indivisible gifts can be allocated to k children. Each child i has a value t_{ij} for gift j, and the total value of a child simply sums t_{ij} over all gifts j awarded to child i. The goal is to find an assignment of gifts that maximizes the value of the *least happy child* (called the fairness). There is also a large body of work in designing both algorithms and mechanisms for fairness maximization, which we overview in Section 4.3 of Chapter 4. We only note here that for certain ranges of k, m, the best known computationally efficient *algorithm* is a $\min\{\tilde{O}(\sqrt{k}), m-k+1\}$-approximation.[6]

Informal Theorem 4. *There is a computationally efficient truthful mechanism for fair allocation of indivisible goods that achieves a* $\min\{\tilde{O}(\sqrt{k}), m-k+1\}$*-approximation.*

Finally, we turn our attention back to revenue maximization, aiming to make use of our framework to design computationally efficient auctions beyond additive agents. Unfortunately, it turns out that maximizing virtual welfare becomes computationally hard quite quickly in such settings. Does this mean that our approach falls short? Or perhaps instead that the original problem we were trying to solve was computationally hard as well. We show that indeed the latter is true, providing an approximation-sensitive[7] reduction from designing algorithms for virtual welfare maximization to designing mechanisms for revenue. Combined with Informal Theorem 1, this brings our reduction full circle, and in some sense shows that our approach is "tight" for revenue. Again making use of our framework, we provide the first unrestricted hardness of approximation result for revenue maximization. Specifically, we show that revenue maximization is computationally hard for even one monotone submodular agent.[8]

Informal Theorem 5. *It is NP-hard to approximately maximize revenue for one agent whose valuation function for subsets of m items is monotone submodular within any* $\text{poly}(m)$ *factor.*

1.5 Tools and Techniques

Our technical contributions come on several fronts. Our first begins with an observation that all mechanism design problems (of the form studied in this thesis) can be solved by a gigantic linear

[6]We use the standard notation $g(n) = \tilde{O}(f(n))$ to mean that $g(n) = O(f(n)\log^c f(n))$ for some constant c.

[7]A reduction is approximation sensitive if whenever the reduction is given black-box access to an α-approximation, the output is an $f(\alpha)$-approximation, for some function f.

[8]A set function f is monotone if $f(S) \leq f(T)$ for all $S \subseteq T$. f is submodular if it satisfies diminishing returns, or formally $f(S \cup T) + f(S \cap T) \leq f(S) + f(T)$ for all S, T.

program that simply stores a variable telling the mechanism exactly what to do on every possible input. Such a solution is of no actual use, either computationally or to learn anything about the structure of the mechanism, but it does provide a starting point for further improvement. From here, we reformulate this program and drastically shrink the number of variables (essentially by underspecifying mechanisms). After this reformulation, our task reduces to obtaining a separation oracle[9] for some convex region related to the original problem, and our focus shifts to purely algorithmic techniques to resolve this.

From here, our next technical contribution is an extension of the celebrated "equivalence of separation and optimization" framework developed by Grötschel, Lovász, and Schrijver and independently Karp and Papadimitriou [GLS81, KP80]. In 1979, Khachiyan showed that one can optimize linear functions computationally efficiently over any convex region given black-box access to a separation oracle for the same region using the ellipsoid algorithm [Kha79]. Remarkably, Grötschel, Lovász, and Shrijver and Karp and Papadimitriou discovered that the converse is true too: one can obtain a computationally efficient separation oracle for any convex region given black-box access to an algorithm that optimizes linear functions over that same region.

Before continuing we should address a frequently raised question: If you can already optimize linear functions, why would you want a separation oracle? Who uses separation oracles for anything besides optimization? One compelling use is optimizing a linear function over the *intersection* of two convex regions. Simply optimizing over each region separately does nothing intelligent towards optimizing over their intersection. But separation oracles for each region separately can be easily composed into a separation oracle for their intersection, which can then be plugged into Khachiyan's ellipsoid algorithm. There are indeed several other uses [GLS81, KP80], but this is the context in which we apply the framework.

Making use of the framework as-is, we are already able to find the revenue-optimal mechanism in multi-dimensional settings with additive agents, but the existing framework is not robust enough to accommodate combinatorially challenging problems that require approximation. To this end, we extend the equivalence of separation and optimization framework to accommodate traditional ap-

[9]A separation oracle for a convex region P is an algorithm SO that takes as input a point $\vec{x}$ and outputs either "yes" if $\vec{x} \in P$ or a hyperplane $(\vec{w}, t)$ such that $\vec{x} \cdot \vec{w} > t \geq \max_{\vec{y} \in P}\{\vec{y} \cdot \vec{w}\}$ otherwise.

proximation algorithms, a new kind of bi-criterion approximation algorithm, and sampling error in Chapter 6. That is, sometimes one can only approximately optimize linear functions over a convex region and would like some meaningful notion of an approximate separation oracle. Unfortunately, when starting from an approximation algorithm, the reduction of [GLS81, KP80] does not produce a separation oracle, and the resulting algorithm may have quite erratic behavior. We call the resulting algorithm a weird separation oracle ("weird" because of the erratic behavior, "separation oracle" because at the very least this algorithm does sometimes say "yes" and sometimes output a hyperplane) and show that weird separation oracles are indeed useful for optimization. Specifically, we show first that running the ellipsoid algorithm with access to a weird separation oracle instead of a true separation oracle results in an algorithm that approximately optimizes linear functions, and second that any point on which the weird separation oracle says "yes" satisfies a strong notion of feasibility (stronger than simply being inside the original convex region). Proving these claims essentially boils down to analyzing the behavior of the ellipsoid algorithm when executed over a *non-convex region*.

Sometimes, however, one can't even approximately optimize linear functions over the desired convex region computationally efficiently, and we must further search for ways to circumvent this computational hardness. In many such cases, while obtaining a traditional approximation algorithm is NP-hard, it may still be computationally feasible to obtain a bi-criterion approximation algorithm. The specific kind of bi-criterion approximation we consider is where the algorithm is first allowed to scale some coordinates of the solution by a multiplicative factor of β before comparing to α times the optimum (whereas a traditional α-approximation algorithm can be viewed as always having $\beta = 1$). Done carelessly, such bi-criterion approximation algorithms can be completely useless for convex optimization. But done correctly, via what we call (α, β)-approximation algorithms, such algorithms can replace traditional approximation algorithms within the equivalence of separation and optimization framework while only sacrificing an additional factor of β in performance. Section 4.2 in Chapter 4 contains a formal definition of such algorithms and more details surrounding this claim.

Finally, it is sometimes the case that even bi-criterion approximation algorithms are computationally hard to design for a convex region, but that algorithms with small additive error can be obtained via sampling. Such additive error isn't small enough to be handled by the original framework of [GLS81, KP80] directly, and the aforementioned techniques apply only to multiplicative error. To this end, we further strengthen the framework to accommodate sampling error in a consistent manner. Section 4.2 in Chapter 4 contains a formal definition of what we mean by sampling error and more details surrounding these claims.

Thus far we have only described our technical contributions towards developing our general framework for reducing mechanism to algorithm design. In order to apply our framework to the problems of makespan and fairness, we develop new bi-criterion approximation algorithms for the perturbed algorithmic problems output by our framework, namely makespan with costs and fairness with costs. These algorithms are based on existing algorithms for the corresponding problems without costs, and require varying degrees of additional insight.

Finally, we establish that our framework for revenue is "tight" in that the reduction from mechanism to algorithm design holds both ways. The difficulty in completing this framework is that virtually nothing is known about the structure of approximately optimal mechanisms, yet establishing computational intractibility requires one to somehow use such mechanisms to solve NP-hard problems. To this end, we define a restricted class of multi-dimensional mechanism design instances that we call *compatible* and show that compatible instances are both restricted enough for us to make strong claims about the structure of approximately optimal mechanisms, yet also rich enough within which to embed NP-hard problems. Making use of this framework, we establish the first unrestricted hardness of approximation result for revenue maximization. Note that prior work has only been successful in establishing hardness of approximation for the optimal deterministic mechanism in settings where the optimal randomized mechanism can be found in polynomial time [Bri08, PP11].

Moreso than any individual technique, we feel that our greatest contribution is this framework itself. Designing complex truthful mechanisms is a *really* challenging task with little, if any, precedent. Even in very simple settings the optimal mechanism is often quite complex, and the design of such mechanisms requires tools that we simply don't have. In contrast, the algorithms community has developed very sophisticated tools for use in the design of complex algorithms, and our framework allows for these tools to be used for the design of truthful mechanisms as well. Similarly, there is little precedent for hardness of approximation results in revenue maximization, while again the algorithms community has developed very sophisticated tools for hardness of approximation in traditional algorithm design. Our framework allows for these tools as well to be used in mechanism design.

1.6 Organization

Chapter 2 provides the necessary background to understand our results on multi-dimensional mechanism design and formally defines the setting we study. Section 2.4 overviews related work and provides context for our multi-dimensional mechanism design results. In Chapter 3, we provide a complete proof of our results on multi-item auctions (namely, a formal statement of Informal Theorem 1). The purpose of separating this result from the rest is to demonstrate the key ideas behind our approach without the technical tools required for full generality.

In Chapter 4, we extend the setting of Chapter 2 to accommodate the full generality of our complete mechanism to algorithm design reduction. Section 4.3 overviews the large body of work on Algorithmic Mechanism Design.

Chapter 5 provides a complete proof of Informal Theorem 2, a reduction from mechanism to algorithm design. The skeleton of the approach is very similar to that of Chapter 3, but more technical tools are required due to the increased generality.

Chapter 6 provides our algorithmic results related to linear programming, specifically extending the celebrated equivalence of separation and optimization to accommodate various types of approximation error. Theorems from this chapter are used in the proofs of results from Chapters 3 and 5. We separate them here because these results are of independent interest outside of mechanism design.

Chapter 7 provides algorithmic results for makespan minimization and fairness maximization that can be leveraged within our framework to yield formal statements of Informal Theorems 3 and 4.

Chapter 8 provides our hardness of approximation result for revenue maximization (namely, a formal statement of Informal Theorem 5). We again derive this result as part of a larger framework that will be useful in showing other new hardness of approximation results. In Chapter 9, we provide conclusions and open problems.

Finally, in Appendix A we provide algorithmic and structural extensions of Border's Theorem. Both results provide an improved structural understanding of the mechanisms we design in Chapter 3 for revenue maximization. While quite important for this setting, we separate them here because there is no meaningful generalization of these results that applies in the setting of our general reduction.

Results are based on joint work with Yang Cai and Constantinos Daskalakis. Chapters 3 and Appendix A are based on [CDW12a]. Chapters 5 and 8 are based on [CDW13b]. Chapter 6 is based on [CDW12b], [CDW13a], and [DW14]. Chapter 7 is based on [DW14].

Chapter 2

Background

Here we provide the necessary background for our results on multi-dimensional mechanism design. We first introduce general concepts from mechanism design in Section 2.1. In Section 2.2, we define formally the setting considered in the following chapter. Section 2.3 provides the necessary preliminaries on linear programming. Finally, Section 2.4 provides an overview of related work on multi-dimensional mechanism design.

2.1 Concepts in Mechanism Design

Mechanism Design Setting. In this thesis, a mechanism design setting consists of a single central designer and k self-interested agents. The designer has to choose some possible outcome x from a set $\mathcal{F}$ of feasible outcomes, and may also charge prices to the agents.

Agent Preferences. Each agent has preferences over the various outcomes, and these preferences are stored in their *type*. One can interpret the type of agent i as a function t_i mapping outcomes in $\mathcal{F}$ to values in $\mathbb{R}$. We use T to denote the set of possible types that agents may have, and assume that $|T|$ is finite. Throughout this thesis, we assume that agents are *quasi-linear* and *risk-neutral*. That is, the utility of an agent of type t_i for the randomized outcome X when paying (a possibly random price with expectation) p is $\mathbb{E}_{x \leftarrow X}[t_i(x)] - p$. From now on, for notational convenience, we just write $t_i(X)$ to mean $\mathbb{E}_{x \leftarrow X}[t_i(X)]$. We assume that agents are *rational*, in that they behave in a way that maximizes utility.

Private Information. Each agent's type is private information known only to that agent. However, the designer and remaining agents have beliefs about each agent's possible type, modeled as a Bayesian prior. That is, the designer and remaining agents believe that agent i's type is sampled from a distribution $\mathcal{D}_i$. We denote by $\vec{t}$ a *type profile* (that is, a vector listing a type for each agent). We also assume that the beliefs over agents' types are independent, and denote by $\mathcal{D} = \times_i \mathcal{D}_i$ the designer's prior over the joint distribution of type profiles.

Mechanisms. In principle, the designer could set up an arbitrary system for the agents to play, involving randomization, multiple rounds of communication, etc. The agents would then use some strategy to participate in the system, and in the end an outcome would be selected and prices would be charged.

Nash Equilibria. How would utility-maximizing agents interact in such a system? Once the strategies of the remaining agents are fixed, each single agent faces an optimization problem: find the strategy maximizing utility. So if an agent knows exactly what strategies will be employed by the remaining agents, she should find the utility-maximizing strategy (called the *best response*), and use it. If each agent employs a strategy that is a best response to the strategies of other agents, then this strategy profile is a *Nash Equilibrium*, and no agent has any incentive to change their strategy. Nash showed that every finite complete information game (and therefore, any corresponding mechanism, assuming that all agents know the types of all other agents) has a Nash Equilibrium [Nas51]. However, due to the uncertainty over preferences of other agents, the Nash Equilibrium is not the right solution concept for this setting.

Bayes-Nash Equilibria. We would still like to model agent behavior via best responses, but we must also take their uncertainty into account. In Bayesian settings (like ours), an agent's behavior isn't just a single strategy, but is instead a mapping from possible types to strategies. Fixing an agent's type, if that agent knows exactly what mapping will be employed by the remaining agents (but still has incomplete information about their types), she should still play a strategy that is a best response. Overloading notation, one can then say a mapping is a best response if it maps every possible type to a strategy that is a best response. If each agent employs a mapping that is a best response to the mappings of other agents, then the collection of mappings is a *Bayes-Nash Equilirbium*, and no agent has any incentive to change their mapping. Every finite incomplete information game has a Bayes-Nash Equilibrium [OR94].

Revelation Principle. Even though we now have a meaningful solution concept with which to predict agent behavior, it's not clear how one should find a Bayes-Nash Equilibrium in an arbitrary mechanism (in fact, it is often computationally hard [CS08, DGP09, CP14]), or further, how one should optimize over the space of all Bayes-Nash Equilibria of all mechanisms. To cope with this, Myerson introduced the *revelation principle* [Mye79, Mye81], which states that every incomplete information game can be simulated by a *direct mechanism.* A direct mechanism simply asks each agent to report a type, then directly selects an outcome (and charges prices) based on the reported types. Given any system, one can imagine assigning a consultant to each agent whose job it is to play that system in a way that maximizes the utility of his agent. The agent reports her type to her consultant so that the consultant knows her preferences, and then the consultants play the resulting system in a Bayes-Nash Equilibrium. Myerson's revelation principle essentially suggests viewing the consultants as part of the system. So the agents simply report a type to the new system, the consultants play the original system in a Bayes-Nash Equilibrium, and some outcome is directly chosen. Furthermore, Myerson showed that it is a Bayes-Nash Equilibrium in this new system for each agent to report their true type. All of this is to say that even though in spirit the designer could design an arbitrarily complicated system and expect agents to behave according to a Bayes-Nash Equilibrium of that system, he may without loss of generality instead design a direct mechanism where truthful reporting is a Bayes-Nash Equilibrium that selects the same outcome (and charges the same prices).

Direct Mechanisms. In our setting, a direct mechanism consists of two functions, a (possibly randomized) allocation rule and a (possibly randomized) price rule, which may be correlated. The allocation rule takes as input a type profile $\vec{t}$ and (possibly randomly) outputs an allocation $A(\vec{t})$. The price rule takes as input a profile $\vec{t}$ and (possibly randomly) outputs a price vector $P(\vec{t})$. When the type profile $\vec{t}$ is reported to the mechanism $M = (A, P)$, the (possibly random) allocation $A(\vec{t})$ is selected and agent i is charged the (possibly random) price $P_i(\vec{t})$.

Truthful Mechanisms. A direct mechanism $M = (A, P)$ is said to be *Bayesian Incentive Compatible (BIC)* if it is a Bayes-Nash Equilibrium for every agent to report truthfully their type. Formally, for all i, $t_i, t_i' \in T$ we must have:

$$\mathbb{E}_{t_{-i} \leftarrow \mathcal{D}_{-i}}[t_i(A(t_i; t_{-i})) - P(t_i, t_{-i})] \geq \mathbb{E}_{t_{-i} \leftarrow \mathcal{D}_{-i}}[t_i(A(t_i'; t_{-i})) - P(t_i', t_{-i})].$$

That is, assuming that all other agents report truthfully their type, each agent's utility is maximized by telling the truth. A direct mechanism is said to be *Individually Rational (IR)* if it is in every agent's best interest to participate in the mechanism, no matter their type. Formally, for all i, $t_i \in T$ we must have:

$$\mathbb{E}_{t_{-i} \leftarrow \mathcal{D}_{-i}}[t_i(A(t_i; t_{-i})) - P(t_i, t_{-i})] \geq 0.$$

Dominant Strategy Incentive Compatibility (DSIC) is a more robust notion of truthfulness than BIC. A direct mechanism is DSIC if it is in each agent's best interest to report truthfully their type, no matter what the other agents choose to report. Formally, for all $i, t_i, t_i', \vec{t}_{-i}$ we must have:

$$t_i(A(t_i; t_{-i})) - P(t_i, t_{-i}) \geq t_i(A(t_i'; t_{-i})) - P(t_i', t_{-i}).$$

BIC and DSIC are both frequently studied notions of truthfulness. DSIC is obviously a more robust solution concept, but BIC allows for a richer set of mechanisms. In other words, maybe the performance of the optimal BIC mechanism is significantly better than that of the optimal DSIC mechanism, and it's unclear which is objectively "better:" If you believe that the agent's will play according to a Bayes-Nash Equilibrium, then the BIC mechanism is better. If you only believe that agents are capable of playing dominant strategies, and that their behavior will be otherwise unpredictable, then the DSIC mechanism is better. In single-dimensional settings, Myerson's characterization also shows that the optimal BIC mechanism is in fact DSIC [Mye81]. But in all settings considered in this thesis, it's unknown how much better the optimal BIC mechanism performs. In such settings, one typically aims to design BIC mechanisms that are competitive with the optimal BIC mechanism. If the designed mechanism happens to be DSIC as well, that is an additional bonus. We restrict attention in this thesis completely to BIC mechanisms, and only discuss DSIC in reference to related work.

Ex-Post Individual Rationality. A mechanism is said to be *ex-post individually rational* if it is in every agent's best interest to participate in the mechanism, no matter their type, no matter the types of the other agents, and no matter the outcome of any random coin tosses used by the mechanism. In this thesis, we focus on designing mechanisms that are just individually rational, but there is a simple reduction turning any individually rational mechanism into one that is ex-post individually rational at no cost (in any setting considered in this thesis).

Observation 1. *Let $M = (A, P)$ be a BIC, IR mechanism. Then there is a BIC, ex-post IR mechanism $M' = (A, P')$ obtaining the same expected revenue as M.*

Proof. For any agent i, and type t_i, let $V_i(t_i)$ denote the expected value obtained by agent i when truthfully reporting type t_i to M (over the randomness in the other agents' types and any randomness in M). Let also $p_i(t_i)$ denote the expected price paid over the same randomness. Define M' to first (possibly randomly) select an outcome $x \in \mathcal{F}$ according to A, then charge agent i price $\frac{p_i(t_i) \cdot t_i(x)}{V_i(t_i)}$.

As M was individually rational, we must have $\frac{p_i(t_i)}{V_i(t_i)} \leq 1$, so M' is ex-post individually rational. It's also clear that the expected payment made by agent i when reporting type t_i is exactly $\mathbb{E}[\frac{p_i(t_i) \cdot t_i(x)}{V_i(t_i)}] = \frac{p_i(t_i)}{V_i(t_i)} \mathbb{E}[t_i(x)] = p_i(t_i)$. Therefore, if M was BIC, so is M'. Finally, notice that M and M' have the same allocation rule and the same expected revenue. □

In light of Observation 1, we will just state and prove throughout this thesis that our mechanisms are individually rational. However, Observation 1 guarantees that all of our mechanisms can be made ex-post IR as well at no cost.

Goal of the Designer. The designer has some objective function in mind that he hopes to optimize, and this objective may depend on the types of the agents, the prices charged, and the outcome selected. For instance, welfare is the collective value of the agents for the outcome selected and can be written as $\sum_i t_i(x)$. Revenue is the collective payments made by the agents to the designer, and can be written as $\sum_i p_i$. Makespan is the maximum value over all agents and can be written as $\max_i\{t_i(x)\}$. Fairness is the minimum value over all agents and can be written as $\min_i\{t_i(x)\}$. Keeping in mind the revelation principle, the goal of the designer is to find the BIC, IR mechanism M that optimizes in expectation the designers objective when agents with types sampled from $\mathcal{D}$ play M truthfully.

Formal Problem Statements. We develop black-box reductions between two problems that we call **B**ayesian **Me**chanism **D**esign (BMeD) and the **G**eneralized **O**bjective **O**ptimization **P**roblem (GOOP). BMeD is a well-studied mechanism design problem. GOOP is a new algorithmic problem that we show has strong connections to BMeD. We provide formal statements of BMeD and GOOP in the following section (and again in Chapter 5 for the general problem).

2.2 Multi-Dimensional Mechanism Design - Additive Agents

Here, we further clarify some mechanism design terms as they apply to this setting and state formally the problem we solve.

Mechanism Design Setting. In this chapter, the central designer has m heterogeneous items for sale to k self-interested agents. The designer may allocate each item to at most one agent, and charge them prices.

Agent Preferences. In this chapter, we further assume that agents are *additive*. That is, agent i has some value t_{ij} for each item j, and the agent's value for any outcome that awards him the set S of items is $\sum_{j \in S} t_{ij}$. We may therefore think of t_i as a vector and will sometimes write $\vec{t_i}$ to emphasize this view.

Private Information. We still assume that each agent's type is sampled independently from a known distribution $\mathcal{D}_i$. We make no assumptions about these distributions. For instance, the value of an agent for items j and j' may be correlated, but the value of agents i and i' for item j are independent. We also use $\mathcal{D}_{ij}$ to denote the marginal of $\mathcal{D}_i$ on item j.

Reduced Forms. It will be helpful to think of mechanisms in terms of their *reduced form* [MR84, Mat84, Bor91, Bor07, MV10, CKM11, HR11]. The reduced form (of a mechanism M) is a function that takes as input an agent i, item j, and type $t_i \in T$ and outputs the probability that agent i receives item j when reporting type t_i to the mechanism M, over any randomness in M and in the other agents' types (assuming they are drawn from $\mathcal{D}_{-i}$. We write $\pi_{ij}^M(t_i)$ to denote the probability that agent i receives item j when reporting type t_i. We often want to think of the reduced form as a vector that simply lists $\pi_{ij}^M(t_i)$ for all i, j, t_i, and write $\vec{\pi}^M$ to emphasize this view. We will also want to treat separately the probabilities seen by agent i when reporting type t_i across all items and denote by $\vec{\pi}_i^M(t_i)$ these probabilities. We will also use $p_i^M(t_i)$ to denote the expected price paid by agent i when reporting type t_i to the mechanism (again over any randomness in M and the other agents' types).

With the reduced form in mind, we may rewrite the mathematical statement of Bayesian Incentive Compatibility as simply $\vec{t_i} \cdot \vec{\pi}_i^M(t_i) - p_i^M(t_i) \geq \vec{t_i} \cdot \vec{\pi}_i^M(t_i') - p_i^M(t_i')$ for all $i, t_i \in T$. We may also rewrite Individual Rationality as simply $\vec{t_i} \cdot \vec{\pi}_i^M(t_i) - p_i^M(t_i) \geq 0$ for all $i, t_i \in T$.

Feasible Reduced Forms. We can think of reduced forms separately from mechanisms, and say that any vector $\vec{\pi} \in \mathbb{R}^{km|T|}$ is a reduced form. We say that a reduced form $\vec{\pi}$ is *feasible* if there exists a feasible mechanism M (i.e. one that awards each item at most once on every profile) such that $\vec{\pi}^M = \vec{\pi}$.

Goal of the Designer. In this chapter, the designer's objective function is revenue.

Formal Problem Statements. Informally, BMeD asks for a BIC, IR mechanism that maximizes expected revenue. GOOP asks for an allocation that awards each item to the highest non-negative bidder. Note that GOOP is trivial to solve in this case, but we use this terminology anyway to make the jump to general objectives easier.

BMeD. INPUT: A finite set of types $T \subseteq \mathbb{R}^m$, and for each agent $i \in [k]$, a distribution $\mathcal{D}_i$ over T. GOAL: Find a feasible (on every profile awards each item at most once) BIC, IR mechanism M that maximizes expected revenue when k agents with types sampled from $\mathcal{D} = \times_i \mathcal{D}_i$ play M truthfully (with respect to all feasible, BIC, IR mechanisms).[1]

GOOP. Given as input $\vec{f} \in \mathbb{R}^{km}$, assign item j to the agent i_j satisfying $i_j = \text{argmax}_i\{f_{ij}, 0\}$.

2.3 Linear Programming

In this section, we provide the necessary preliminaries on linear programming for the subsequent results in Chapter 3.

Computation and Bit Complexity. As our solutions make use of the ellipsoid algorithm, their running time necessarily depends on the *bit complexity* of the input. We say that the bit complexity of a rational number r is b if r can be written as $r = x/y$, where x and y are both b-bit integers. We say that the bit complexity of a vector $\vec{x} \in \mathbb{R}^d$ is b if each coordinate of $\vec{x}$ has bit complexity b.

Hyperplanes and Halfspaces. A *hyperplane* in $\mathbb{R}^d$ corresponds to a direction $\vec{w} \in \mathbb{R}^d$ and value c and is the set of points $\{\vec{x}\,|\vec{x} \cdot \vec{w} = c\}$. A *halfspace* contains all points lying to one side of a hyperplane, and can be written as $\{\vec{x}\,|\vec{x} \cdot \vec{w} \leq c\}$.

Closed Convex Regions. A set of points P is *convex* if for all $\vec{x}, \vec{y} \in P$, $z\vec{x} + (1-z)\vec{y} \in P$, for all $z \in [0,1]$. P is *closed* if it contains all its limit points.[2] A well-known fact about closed, convex regions is stated below.

[1]By "find a mechanism" we formally mean "output a computational device that will take as input a profile of types $\vec{t}$ and output (possibly randomly) a feasible outcome and a price to charge each agent." The runtime of this device is of course relevant, and will be addressed in our formal theorem statements.

[2]$\vec{x}$ is a limit point of P if for all $\epsilon > 0$, the ball of radius ϵ centered at $\vec{x}$ contains a point in P.

Fact 1. *A region $P \in \mathbb{R}^d$ is closed and convex if and only if P can be written as an intersection of halfspaces. That is, there exists an index set $\mathcal{I}$ such that $P = \{\vec{x} \,|\, \vec{x} \cdot \vec{w}_i \le c_i,\ \forall i \in \mathcal{I}\}$.*

Separation Oracles. A *separation oracle* for a closed convex region P is an algorithm that takes as input a point $\vec{y}$ and outputs "yes" if $\vec{y} \in P$, or a *separating hyperplane* otherwise. A separating hyperplane is a direction $\vec{w}$ and value c such that $\vec{x} \cdot \vec{w} \le c$ for all $\vec{x} \in P$ but $\vec{y} \cdot \vec{w} > c$. Fact 1 above guarantees that every closed convex region admits a separation oracle.

Corners. Throughout this thesis, we use the term *corner* to refer to non-degenerate extreme points of a closed convex region. In other words, $\vec{y}$ is a corner of the d-dimensional closed convex region P if $\vec{y} \in P$ and there exist d linearly independent directions $\vec{w}_1, \ldots, \vec{w}_d$ such that $\vec{x} \cdot \vec{w}_i \le \vec{y} \cdot \vec{w}_i$ for all $\vec{x} \in P, 1 \le i \le d$. Two well-known facts about corners of closed convex regions are stated below.

Fact 2. *If P is a closed convex region and $\vec{y}$ is a corner of P, then there is a corresponding direction $\vec{w}$ such that $\vec{y} \cdot \vec{w} > \vec{x} \cdot \vec{w}$ for all $\vec{x} \in P - \{\vec{y}\}$.*

Fact 3. *Let $P = \{\vec{x} \,|\, \vec{x} \cdot \vec{w}_i \le c_i, \forall i \in \mathcal{I}\}$ be a d-dimensional closed convex region such that c_i and $\vec{w}_i$ has bit complexity at most b for all $i \in \mathcal{I}$. Then every corner of P has bit complexity at most* $\text{poly}(b, d)$.

Ellipsoid Algorithm. Khachiyan's ellipsoid algorithm optimizes linear functions over any closed convex region P in polynomial time with black-box access to a separation oracle for P. This is stated formally below. In Theorem 1 below, and throughout this thesis, we use the notation $\text{runtime}_A(b)$ to denote an upper bound on the running time of algorithm A on inputs of bit complexity b.

Theorem 1. *[Ellipsoid Algorithm for Linear Programming] Let P be a closed convex region in $\mathbb{R}^d$ specified via a separation oracle SO, and $\vec{c} \cdot \vec{x}$ be a linear function. Assume that all $\vec{w}$ and c, for all separation hyperplanes $\vec{w} \cdot \vec{x} \le c$ possibly output by SO, and all $\vec{c}$ have bit complexity b. 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* $\text{poly}(d, b)$.

2. *The algorithm will solve the linear program in time* $\text{poly}(d, \ell, \text{runtime}_{SO}(\text{poly}(d, b)))$.

3. *The output optimal solution is a corner of P.*

Equivalence of Separation and Optimization. We make use of a linear programming framework colloquially called the equivalence of separation and optimization. We state the theorem below, and provide a proof in Section 6.1 of Chapter 6 in order to acclimate the reader with the relevant techniques.

Theorem 2. *([GLS81, KP80]) Let P be any d-dimensional closed convex region and let $\mathcal{A}$ be an algorithm that optimizes linear functions over P. That is, $\mathcal{A}$ takes as input a d-dimensional vector $\vec{w}$ and outputs $\vec{x} \in$* $\text{argmax}_{\vec{x} \in P}\{\vec{x} \cdot \vec{w}\}$. *Then there exists a separation oracle SO for P such that* $\text{runtime}_{SO}(b) = \text{poly}(d, b, \text{runtime}_{\mathcal{A}}(\text{poly}(d, b)))$.

Decomposition Algorithm. In Chapter 3, we will also make use of the following theorem, stating that with black-box access to a separation oracle for a closed convex region P, one can decompose any $\vec{y} \in P$ into a convex combination of corners of P in polynomial time. The algorithm is folklore knowledge, but we prove Theorem 3 in Appendix B for completeness.

Theorem 3. *Let $P = \{\vec{x} \,|\, \vec{x} \cdot \vec{w}_i \leq c_i, \forall i \in \mathcal{I}\}$ be a d-dimensional closed convex region specified via a separation oracle SO. Let also $\vec{y}$ be any point in P, and further assume that $\vec{y}$, c_i and $\vec{w}_i$ have bit complexity at most b for all $i \in \mathcal{I}$. Then $\vec{y}$ can be written as a convex combination of at most $d + 1$ corners of P in time* $\text{poly}(b, d, \text{runtime}_{SO}(\text{poly}(b, d)))$.

2.4 Related Work

There is significant prior work in the field of multi-dimensional mechanism design, from both Economists and Computer Scientists. We overview the most relevant related work addressing both the structural and computational aspects of the problem below.

Structural. As stated before, our understanding of revenue maximization in single-dimensional settings is quite good. Results of Myerson, and Riley and Zeckhauser show that the optimal mechanism is deterministic [Mye81, RZ83]. Myerson further characterizes the optimal mechanism as the one maximizing virtual welfare. Bulow and Roberts later interpreted Myerson's virtual values as a marginal contribution to revenue [BR89]. In other words, one can view Myerson's single-item auction as awarding the item to the agent with the largest marginal contribution to revenue, fitting a familiar theme from Microeconomics.

Moving beyond single-dimensional settings, very little is known about multi-dimensional auctions in completely unrestricted environments. It is folklore knowledge that the optimal mechanism can be found via a large linear program when the designer's prior has finite support [BCKW10, Voh11]. In the case of infinite support, Rochet and Choné show that the optimal mechanism is the solution of a differential equation after a complex "sweeping" technique [RC98]. Rochet also shows that every truthful mechanism satisfies a property called *cyclic monotonicity* [Roc87]. Cyclic monotonicity is a multi-dimensional analogue of monotonicity, and comes into play in Chapter 8. Recently, Daskalakis, Deckelbaum, and Tzamos have developed a duality framework for a single additive buyer, showing that a "dual" to the revenue-optimal mechanism can be found by solving an optimal transport problem [DDT13, DDT14].

While the results of the previous paragraph are certainly of interest, they are also quite far from a characterization as strong as Myerson's. To this end, much work has also been devoted to providing a stronger structural analysis in special cases. One such direction indentifies certain "multi-dimensional hazard rate" conditions (similar to the one-dimensional condition of regularity) under which the optimal mechanism has more structure [MM88]. Subject to the specifics of the setting and these conditions, the optimal mechanism might be deterministic [MV06, Pav11], involve limited randomness [TW14], or be the solution to a simple differential equation [Arm96]. As overviewed in Section 1.2.1 in Chapter 1, numerous examples bear witness to the need for such restrictions in order to obtain these stronger structural results [Pav06, BCKW10, DDT13, DDT14, HN12, HR12, Pav11, Tha04].

Another direction aims to classify further the space of truthful deterministic mechanisms. To this end, Roberts shows that under certain assumptions on the space of possible types, every DSIC deterministic mechanism is an *affine maximizer* [Rob79]. That is, every DSIC mechanism awards to each agent a non-negative multiplier λ_i, and selects the outcome that maximizes the scaled welfare, $\text{argmax}_{x \in \mathcal{F}}\{\sum_i \lambda_i t_i(x)\}$. Prior to our work, no analogue of Roberts' Theorem was known for either randomized mechanisms or BIC mechanisms [Mis12]. Our structural results in Chapters 3 and 5 can therefore also be seen as an extension of Roberts' Theorem both to randomized and BIC mechanisms without requiring any assumption on the type space. We show that every BIC mechanism is "equivalent" to a virtual welfare maximizer.

Computational. Numerous computationally efficient mechanisms have been developed in recent years with various approximation and runtime guarantees in various settings. Before continuing, we briefly discuss possible runtime guarantees, which have two flavors. The first guarantee is simply that the mechanism can be found and implemented in time $\text{poly}(|T|, k, m)$. For many settings (in particular, the settings considered in this thesis), this is the natural description size of the input. Some recent results require the additional assumption that each $\mathcal{D}_i$ is a product distribution. In this case, the natural description of the input is not to list explicitly the support of each $\mathcal{D}_i$ and the corresponding probabilities, but instead to list the support of each $\mathcal{D}_{ij}$ and the corresponding probabilities. In describing prior work, when we say "polynomial time," we mean $\text{poly}(|T|, k, m)$ if no assumptions are made on the input distribution, or $\text{poly}(\max_{i,j}\{|\mathcal{D}_{ij}|\}, k, m)$ if the input is required to be a product distribution. All mechanisms described in this section run in polynomial time.

We also note that *all* settings considered in prior work model additive agents with combinatorial constraints on which agents can simultaneously receives which items. This includes, for instance, settings with *unit-demand* or *n-demand* agents,[3] where we can instead model agents as additive and constrain the designer to never allocate a unit-demand agent more than one item (or an n-demand agent more than n items). Another special case of this setting is what are called *service constrained environments* [AFH+12, AFHH13], where agents have multi-dimensional preferences, but feasibil-

[3]An agent is unit demand if she has a value v_i for item i and value $\max_{i \in S}\{v_i\}$ for a set S of items. An agent is n-demand if she has a value v_i for item and value $\max_{S' \subseteq S, |S'| \leq n}\{\sum_{i \in S'} v_i\}$ for a set S of items.

ity only constrains which agents can simultaneously be served and importantly *doesn't* constrain which items they receive. With the exception of [KW12], all prior work is limited to studying n-demand agents or service constrained environments, whereas our techniques apply to arbitrary combinatorial constraints. Furthermore, our reduction provides the first algorithmic framework with which to study multi-dimensional settings beyond additive agents (e.g. agents whose values for sets of items are submodular, gross substitutes, subadditive, etc.).

Still, prior work has made great progress for the settings considered. One such setting is that of many unit-demand agents whose values for each of the items is drawn independently. Although this setting is multi-dimensional, Chawla, Hartline, and Kleinberg relate it to a single-dimensional setting where virtual welfare analysis can be used to design a pricing scheme that obtains a 3-approximation for a single agent [CHK07]. These techniques were further extended to yield constant-factor approximations in settings with many agents and combinatorial constraints on which agents may simultaneously receive which items [CHMS10, CMS10, KW12, AKW14]. Furthermore, all mechanisms designed via this approach are DSIC.

Another line of work studies settings with a constant number of n-demand agents whose values for each item are drawn independently from one-dimensional distributions that satisfy the *monotone hazard rate* condition.[4] Cai and Daskalakis develop extreme value theorems for such distributions, and use them to find a near-optimal pricing scheme for a single unit-demand agent (specifically, they develop a Polynomial Time Approximation Scheme (PTAS) [CD11].[5] These techniques were later extended to obtain near-optimal mechanisms in settings with a constant number of additive agents, or many i.i.d. additive agents [CH13] as well as a constant number of n-demand agents whose values for the items are all i.i.d. [DW12].

Still another line of work makes use of linear programming and linear programming relaxations. Constant-factor approximations are developed via linear programming relaxations for additive agents in [Ala11, BGGM10]. Our work is most similar to that of Alaei et. al. and Bhalgat et. al. who also solve linear programs using the reduced form to find optimal mecha-

[4]A one-dimensional distribution satisfies the monotone hazard rate condition if $\frac{1-F(x)}{f(x)}$ is monotonically decreasing.

[5]A collection of algorithms parameterized by ϵ is a PTAS if for any desired constant ϵ, the algorithm corresponding to ϵ yields a $(1-\epsilon)$-approximate solution in polynomial time (but the runtime may depend exponentially or worse on $1/\epsilon$).

nisms [AFH+12, AFHH13, BGM13]. The work of Alaei et. al. was done independently of ours, and studies service constrained environments. The work of Bhalgat et. al. was subsequent to our work on multi-dimensional mechanism design and extended our framework to accommodate a different linear programming algorithm (namely, Multiplicative Weights Updates) and various other economic constraints, such as envy-freeness or ex-post budget constraints.

Finally, another line of work aims to design extremely simple approximately optimal auctions, following the similar work of Hartline and Roughgarden in single-dimensional settings [HR09]. Hart and Nisan showed that when a single additive agent's values for each item are independent, that selling the items separately yields a poly-logarithmic approximation (in the number of items) [HN12]. Li and Yao later improved this to a logarithmic approximation [LY13]. Babaioff et. al. extend the logarithmic approximation to many bidders, and also show that either selling the items separately or bundling them all together yields a constant-factor approximation for a single agent [BILW14].

In comparison to this related work, our mechanisms developed in Chapter 5 apply to *every* setting previously studied, providing a BIC mechanism that is a $(1 - \epsilon)$ approximation in time poly$(1/\epsilon,\ |T|, m, k)$. In the case of additive buyers, we provide an exactly optimal solution in polynomial time in Chapter 3. Still, our results do not subsume all prior work: some previously designed mechanisms are DSIC, some obtain a better running time in the case of product distributions, and some have more structure than those we design. Our work constitutes a major contribution in two ways. First, we extend greatly the capability of the state-of-the-art to include numerous important settings where previously no results were known. Second, we provide a unifying framework and generally applicable techniques with which to tackle mechanism design problems in unrestricted settings, whereas previous techniques were highly specialized, catering to limited settings.

Reduced Forms. A key technical contribution enabling our results is an algorithm for determining when a given reduced form is feasible. This question was already studied in single-item settings beginning with work of Maskin and Riley, and Matthews [MR84, Mat84]. The notable result in this line of work is Border's Theorem, which provides compelling necessary and sufficient conditions for feasible reduced forms [Bor91, Bor07]. Subsequent work provided other formulations and applications of Border's Theorem [MV10, CKM11, HR11]. Importantly however, prior work did *not* yield a computationally efficient algorithm to check whether or not Border's conditions hold for a given reduced form. We discuss further Border's Theorem in Appendix A and provide such an algorithm there. We also note that a similar algorithm was discovered independently by Alaei et. al. [AFH$^+$12]. Neither our algorithm in Appendix A nor that of [AFH$^+$12] generalizes beyond single-dimensional settings. For this reason, we present a different algorithm based on the equivalence of separation and optimization in the following chapter that generalizes naturally to multi-dimensional settings.

Chapter 3

Additive Agents

In this chapter, we characterize the optimal multi-dimensional mechanism with additive agents as a distribution over virtual-welfare maximizers and show how to find it computationally efficiently. More specifically, we show that the revenue-optimal auction takes the following form:

1. Each agent reports their value for each item.
2. These values are transformed into virtual values. This transformation is randomized and pre-computed by a linear program.
3. Each item is awarded to the agent with the highest non-negative virtual value (or no one).
4. Payments are charged to ensure that the mechanism is truthful. These payments are also pre-computed by a linear program.

This is stated formally in Theorem 4 below.

Theorem 4. *Let b be an upper bound on the bit complexity of t_{ij} and $Pr[t_i \leftarrow \mathcal{D}_i]$ for all t_i. BMeD can be solved in time* $\text{poly}(|T|, m, k, b)$*, and the mechanism found can be implemented in time* $\text{poly}(|T|, m, k, b)$ *as well. Furthermore, the mechanism output will be a distribution over virtual-welfare maximizers.*

Here is a brief overview of our approach: We begin by writing a linear program using the reduced form of an auction. It is easy to compute the expected revenue of a mechanism with reduced form $(\vec{\pi}, \vec{p})$, as the expected revenue is just a linear function of $\vec{p}$. Furthermore, it is easy to check whether or not a mechanism with reduced form $(\vec{\pi}, \vec{p})$ is truthful: because agents are additive, quasi-linear and risk-neutral, their expected value is also a linear function of $(\vec{\pi}, \vec{p})$ (see Section 2.2 in Chapter 2). The remaining challenge is determining if a reduced form is *feasible* or not. In other words, we still need to ensure that any reduced form we accept not only represents a truthful mechanism, but also one that doesn't allocate any item more than once on any profile.

Explicitly writing linear constraints guaranteeing feasibility of a reduced form is actually quite challenging, so we take a different approach. Instead of writing the constraints explicitly, we aim to design a separation oracle for the space of feasible reduced forms. We then obtain such a separation oracle using an approach colloquially called "the equivalence of separation and optimization." The combined results of Khachiyan, and Grötschel, Lovász and Schrijver, and Karp and Papadimitriou claim the following: one can obtain a separation oracle for a closed convex region P in polynomial time if and only if one can optimize linear functions over P in polynomial time. The only if direction is due to Khachiyan's Ellipsoid algorithm [Kha79], and the if direction is due to independent work of Grötschel, Lovász and Schrijver [GLS81], and Karp and Papadimitriou [KP80]. For our purposes, this means that we can find the revenue-optimal reduced form so long as we can optimize linear functions over the space of feasible (but not necessarily truthful) reduced forms.

Finally, we show that linear functions over the space of feasible reduced forms essentially correspond to a virtual welfare computation, for appropriately defined virtual transformations. So optimizing linear functions over the space of feasible reduced forms exactly corresponds to maximizing virtual welfare, without any truthfulness constraints. And maximizing virtual welfare in this setting is trivial: just give every item to the agent with the highest non-negative virtual value, if one exists.

Putting everything together, we start with a simple algorithm for maximizing virtual welfare. This algorithm allows us then to optimize linear functions over the space of feasible reduced forms, which then allows us to get a separation oracle for the space of feasible reduced forms using the equivalence of separation and optimization. Finally, this separation oracle allows us to optimize over the space of reduced forms that are both feasible and truthful to find the one that optimizes expected revenue.

3.1 A Linear Program

In this section we write a linear program to find the reduced form of the revenue optimal mechanism. We begin with a linear program that is exponentially large, but easy to understand in Figure 3-1.

Variables:

- $\pi_{ij}(t_i)$, for $1 \le i \le k$, $1 \le j \le m$, $t_i \in T$, denoting the probability that item j is awarded to agent i when reporting type t_i.
- $p_i(t_i)$, for $1 \le i \le k$, $t_i \in T$, denoting the expected price paid by agent i when reporting type t_i.
- $\phi_{ij}(\vec{t})$, for $1 \le i \le k$, $1 \le j \le m$, $\vec{t} \in \times_{i\in[k]} T$, denoting the probability that item j is awarded to agent i on profile $\vec{t}$.

Constraints:

- $\sum_j \pi_{ij}(t_i) \cdot t_{ij} - p_i(t_i) \ge \sum_j \pi_{ij}(t_i') \cdot t_{ij} - p_i(t_i')$, for $1 \le i \le m$, $t_i, t_i' \in T$, guaranteeing that the reduced form $(\vec{\pi}, \vec{p})$ is truthful.
- $\sum_j \pi_{ij}(t_i) \cdot t_{ij} - p_i(t_i) \ge 0$, for $1 \le i \le m$, $t_i \in T$, guaranteeing that the reduced form $\vec{\pi}$ is individually rational.
- $\sum_i \phi_{ij}(\vec{t}) \le 1$, for $1 \le j \le m$, $\vec{t} \in \times_{i\in[k]} T$, guaranteeing that the mechanism is feasible (that each item is awarded at most once on every profile).
- $Pr[t_i' \leftarrow \mathcal{D}_i] \cdot \pi_{ij}(t_i') = \sum_{\vec{t}|t_i=t_i'} Pr[\vec{t} \leftarrow \mathcal{D}] \cdot \phi_{ij}(\vec{t})$, for all $i, j, t_i' \in T$ guaranteeing that the reduced form is computed correctly.

Maximizing:

- $\sum_i \sum_t Pr[t \leftarrow \mathcal{D}_i] \cdot p_i(t_i)$, the expected revenue of a mechanism with reduced form $(\vec{\pi}, \vec{p})$.

Figure 3-1: A linear programming formulation to find the revenue-optimal mechanism.

Observation 2. *Let b be an upper bound on the bit complexity of t_{ij} and $Pr[t_i \leftarrow \mathcal{D}_i]$ for all t_i. The linear program of Figure 3-1 finds explicitly the revenue-optimal mechanism. Furthermore, this*

LP can be solved in time $\text{poly}(m, |T|^k, b)$.

Proof. It is clear that there is a one to one correspondence between feasible, BIC mechanisms and solutions to the LP in Figure 3-1. Given any solution to the LP, simply award item j to agent i with probability $\phi_{ij}(\vec{t})$ on profile $\vec{t}$. The linear constraints guarantee that this is feasible, because the probabilities sum to at most 1, and truthful, because the linear constraints for BIC and IR are satisfied. Given any BIC, IR mechanism, one can similarly construct a feasible solution to the LP. Therefore, the solution output by the LP corresponds to the revenue-optimal mechanism. That the LP can be solved in the desired runtime is an immediate corollary of Khachiyan's ellipsoid algorithm (Theorem 1 in Chapter 2). □

From here, we reformulate the linear program of Figure 3-1 to reduce the number of variables. Essentially, we would like to remove the $\phi_{ij}(\vec{t})$ variables completely, as there are exponentially many of them, and keep only the variables corresponding to the reduced form. But in doing so, we lose the ability to explicitly write linear constraints guaranteeing that the reduced form corresponds to an actual mechanism at all, requiring us to replace these constraints with a separation oracle.

Observation 3. *Let b be an upper bound on the bit complexity of t_{ij} and $Pr[t_i \leftarrow \mathcal{D}_i]$ for all t_i. The LP of Figure 3-2 outputs the reduced form of the revenue-optimal mechanism. Furthermore, this LP can be solved in time* $\text{poly}(|T|, k, m, b, \text{runtime}_{SO}(\text{poly}(|T|, k, m, b)))$*, where SO is a separation oracle determining feasibility of reduced forms.*

The proof of Observation 3 immediately follows from Observation 2 and Khachiyan's ellipsoid algorithm (Theorem 1 in Chapter 2).

3.2 The Space of Feasible Reduced Forms

By Observation 3, we can solve BMeD as long as we can find a computationally efficient separation oracle for the space of feasible reduced forms (which we denote by $F(\mathcal{D})$). Our approach will be to use the equivalence of separation and optimization (Theorem 2 in Section 2.3 of Chapter 2). Based on this equivalence, the goal of this section is just to develop a computationally efficient algorithm

Variables:

- $\pi_{ij}(t_i)$, for $1 \leq i \leq m$, $1 \leq j \leq n$, $t_i \in T$, denoting the probability that item j is awarded to agent i when reporting type t_i.
- $p_i(t_i)$, for $1 \leq i \leq m$, $t_i \in T$, denoting the expected price paid by agent i when reporting type t_i.

Constraints:

- $\sum_j \pi_{ij}(t_i) \cdot t_{ij} - p_i(t_i) \geq \sum_j \pi_{ij}(t'_i) \cdot t_{ij} - p_i(t'_i)$, for $1 \leq i \leq m$, $t_i, t'_i \in T$, guaranteeing that the reduced form $(\vec{\pi}, \vec{p})$ is truthful.
- $\sum_j \pi_{ij}(t_i) \cdot t_{ij} - p_i(t_i) \geq 0$, for $1 \leq i \leq m$, $t_i \in T$, guaranteeing that the reduced form $\vec{\pi}$ is individually rational.
- $(\vec{\pi}, \vec{p})$ is feasible. **A separation oracle is needed for this.**

Maximizing:

- $\sum_i \sum_t Pr[t \leftarrow \mathcal{D}_i] \cdot p_i(t_i)$, the expected revenue of a mechanism with reduced form $(\vec{\pi}, \vec{p})$.

Figure 3-2: A reformulated linear program to find the revenue-optimal mechanism.

that can optimize linear functions over $F(\mathcal{D})$. Before beginning, we provide some examples of feasible and infeasible reduced forms below in Section 3.2.1. Our proof begins immediately after in Section 3.2.2.

3.2.1 Examples of Reduced Forms

In this section we provide several examples of reduced forms with two agents and one item. The purpose is to help familiarize the reader with the notion of a feasible reduced form. To make the presentation more concise, we state here that in all examples each agent's distribution is uniform over her type space.

Example 1. Each agent has two possible types, A or B. $\pi_1(A) = 1$, $\pi_1(B) = 0$. $\pi_2(A) = 1/2$, $\pi_2(B) = 1/2$. This reduced form is feasible. Consider the mechanism that awards the item to agent 1 whenever she reports type A, and agent 2 otherwise. If agent 1's type is A, she receives the item with probability 1. Agent 2 receives the item whenever agent 1's type is B, no matter what she reports. This occurs with probability $1/2$.

Example 2. Each agent has two possible types, A or B. $\pi_1(A) = 1$, $\pi_1(B) = \epsilon$. $\pi_2(A) = 1/2$, $\pi_2(B) = 1/2$. This reduced form is infeasible. Let $X_i(t_i)$ be the indicator random variable for the event that agent i's type is t_i *and* agent i wins the item. Then $X_1(A) = 1/2$, $X_1(B) = \epsilon/2$, $X_2(A) = 1/4$, and $X_2(B) = 1/4$. The expected number of items given away by any mechanism with this reduced form is therefore $1 + \epsilon/2$. As there is only one item to give away, it's clear that no feasible mechanism can possibly match this reduced form.

A tempting conjecture following Example 2 is that perhaps any reduced form that awards at most one item in expectation is feasible. Example 3 below shows that this is not the case.

Example 3. Each agent has two possible types, A or B. $\pi_1(A) = 1$, $\pi_1(B) = 0$. $\pi_2(A) = 1/2 + \epsilon$, $\pi_2(B) = 0$. In order to possibly be feasible, we'd need to give agent 1 the item whenever her type is A. But now agent 2 can only get the item when agent 1's type is B, which happens with probability $1/2$. So no feasible mechanism can possibly match this reduced form, and it is infeasible. Note also that the expected number of items given away is $3/4 + \epsilon$.

Example 4. Each agent has three possible types, A, B, or C. $\pi_1(A) = 5/6$, $\pi_1(B) = 2/3$, $\pi_1(C) = 0$. $\pi_2(A) = 5/6$, $\pi_2(B) = 1/3$, $\pi_2(C) = 1/3$. This reduced form is feasible. Consider the mechanism with the following behavior: First, award the item to any agent whose type is A no matter what, breaking ties uniformly at random. Next, award the item to any agent whose type is B, breaking ties in favor of agent 1. If both types are C, award the item to agent 2. One can check that this mechanism matches the given reduced form, and therefore the reduced form is feasible.

Example 5. Each agent has three possible types, A, B, or C. $\pi_1(A) = 5/6$, $\pi_1(B) = 2/3$, $\pi_1(C) = 0$. $\pi_2(A) = 5/6$, $\pi_2(B) = 1/3 + \epsilon$, $\pi_2(C) = 0$. This reduced form is infeasible. Define again indicator random variables $X_i(t_i)$ for the event that agent i's type is t_i and she receives the item. Then the expected number of items awarded to an agent whose type is either A or B is equal to

$X_1(A) + X_1(B) + X_2(A) + X_2(B) = 8/9 + \epsilon/3$. As any feasible mechanism can only award one item to an agent with type A or B when *some* agent has type A or B, this value should be less than the probability that some agent has type A or B, which is $8/9$. Therefore, this reduced form is infeasible.

We continue discussing single-item reduced forms in Appendix A. Border developed necessary and sufficient conditions for a reduced form to be feasible [Bor91, Bor07]. We present two new extensions of Border's Theorem in Appendix A as well, one algorithmic and the other structural. We conclude this section by noting that the examples above show that reasoning about the feasibility of reduced forms even in the case of a single item is non-trivial, motivating the series of works resolving this question [MR84, Mat84, Bor91, Bor07, CKM11, HR11, AFH$^+$12, CDW12a].

3.2.2 Understanding the Space of Feasible Reduced Forms

The examples of Section 3.2.1 above motivates the use of non-trivial algorithmic techniques to determine the feasibility of reduced forms. Because there is no interaction between items in terms of feasibility, we could try to make use of previous work (i.e. Border's Theorem [Bor91, Bor07]). Border's theorem states that a reduced form is feasible if and only if it satisfies a list of linear constraints, but unfortunately this list is exponentially long. We present in Appendix A a shorter list of only *linearly many* Border constraints that are equivalent to Border's original list. In other words, a reduced form satisfies every constraint in Border's original list if and only if it satisfies every constraint in our shorter list. Making use of this list would then provide us with a computationally efficient separation oracle for this setting. However, we choose to present here a different proof based on the equivalence of separation and optimization because these ideas generalize to far more complex settings whereas the approach based on Border's theorem does not. We begin now with some helpful structural facts about $F(\mathcal{D})$.

Observation 4. *An allocation rule is feasible if and only if it is a distribution over feasible deterministic allocation rules.*

Proof. For any feasible allocation rule M, and any type profile $\vec{t}$, the (possibly randomized) allocation $M(\vec{t})$ is a distribution over feasible deterministic allocations. So let $M(\vec{t})$ sample the deterministic allocation $A_z(\vec{t})$ with probability $q_z(\vec{t})$. Then $M(\vec{t})$ can be implemented by uniformly sampling x from $[0, 1]$ and selecting $A_z(\vec{t})$ iff $\sum_{j<z} q_j(\vec{t}) < x \leq \sum_{j\leq z} q_j(\vec{t})$. So for $y \in [0, 1]$ let $M^{(y)}$ denote the deterministic allocation rule that on profile $\vec{t}$ implements the deterministic allocation selected by $M(\vec{t})$ 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 1. *$F(\mathcal{D})$ is a closed convex region.*

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. We claim that $F(\mathcal{D})$ 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 4, M is a distribution over deterministic allocation rules, sampling M_z with probability q_z. Therefore, if $\vec{\pi}_z$ denotes the reduced form of M_z, we must have $\vec{\pi} = \sum_z q_z\vec{\pi}_z$, so $\vec{\pi}$ is in the convex hull of S. Similarly, if a reduced form $\vec{\pi}$ satisfies $\vec{\pi} = \sum_z q_z\vec{\pi}_z$, where $\vec{\pi}_z$ is the reduced form of a deterministic allocation rule M_z for all z, the allocation rule that selects M_z with probability q_z implements $\vec{\pi}$. So the space of feasible reduced forms is exactly the convex hull of S, and is therefore closed and convex. □

Now that we know that $F(\mathcal{D})$ is a closed convex region, we want to look at the corners by examining, for any $\vec{w}$, the feasible allocation rule whose reduced form maximizes $\vec{\pi} \cdot \vec{w}$. Lemma 1 and Corollary 1 characterize the corners of $F(\mathcal{D})$.

Lemma 1. *Let M be a mechanism with reduced form $\vec{\pi}$ (computed with respect to the distribution $\mathcal{D}$), and $\vec{w}$ be any direction. Then $\vec{\pi}\cdot\vec{w}$ is exactly the expected virtual welfare of M, when the virtual value of agent i with type t_i for item j is $w_{ij}(t_i)/Pr[t_i \leftarrow \mathcal{D}_i]$.*

Proof. The proof is straight-forward once we correctly interpret $\vec{\pi} \cdot \vec{w}$. Expanding the dot product, we see that:

$$\begin{aligned}\vec{\pi}\cdot\vec{w} &= \sum_i \sum_j \sum_{t_i\in T} \pi_{ij}(t_i) w_{ij}(t_i)\\ &= \sum_i \sum_j \sum_{t_i\in T} \Pr[t_i \leftarrow \mathcal{D}_i]\pi_{ij}(t_i)\frac{w_{ij}(t_i)}{\Pr[t_i \leftarrow \mathcal{D}_i]}.\end{aligned}$$

If the "virtual value" of awarding item j to agent i when her reported type is A is $\frac{w_{ij}(t_i)}{\Pr[t_i\leftarrow\mathcal{D}_i]}$, then the last line is exactly the expected virtual welfare of an allocation rule whose reduced form is $\vec{\pi}$. This is because the term sums over all agents i, all types t_i and all items j the probability that agent i has type t_i times the probability that agent i gets item j conditioned on this report, times the virtual value of receiving item j with this type. □

Corollary 1. *Every corner $\vec{\pi}$ of $F(\mathcal{D})$ has a corresponding (deterministic) virtual transformation such that the mechanism implementing $\vec{\pi}$ maximizes virtual welfare under this transformation.*

Proof. As $\mathcal{F}(\mathcal{D})$ is a closed convex region, every corner $\vec{\pi}$ of $F(\mathcal{D})$ has a corresponding direction $\vec{w}$ (Fact 2 of Section 2.3 in Chapter 2) such that $\vec{\pi} = \text{argmax}_{\vec{x}\in F(\mathcal{D})}\{\vec{x}\cdot\vec{w}\}$. By Lemma 1, this implies that $\vec{\pi}$ is the reduced form with the maximum virtual welfare (under the specific transformation corresponding to $\vec{w}$)) over all feasible reduced forms. Therefore, the mechanism implementing $\vec{\pi}$ necessarily maximizes virtual welfare under this transformation. □

Now that we know the structure of the corners of $F(\mathcal{D})$, we turn back to computation and show that we can optimize linear functions over $F(\mathcal{D})$ computationally efficiently. Until this point, we have made use of the fact that agents are additive, but not of the specific nature of the feasibility constraints. Proposition 1, Lemma 1, and Corollary 1 still hold true even if there are combinatorial constraints on which agents can simultaneously receive which items. However, computing exactly the reduced form of a virtual welfare maximizer is non-trivial in the face of combinatorial constraints and requires new tools in linear programming that we develop in Section 6.3 of Chapter 6. It is for this reason that we choose to treat this case separately from the completely general settings of Chapter 5. In Corollary 2 below, note that we are indeed making use of the simple nature of feasibility constraints in this setting (that each item can be given to any agent, but at most one).

Corollary 2. *There is a polynomial time algorithm takes as input a direction $\vec{w}$ and outputs* $\text{argmax}_{\vec{\pi}\in F(\mathcal{D})}\{\vec{\pi}\cdot\vec{w}\}$. *If b upper bounds the bit complexity of $\vec{w}$ and $Pr[t_i \leftarrow \mathcal{D}_i]$ for all i, then the algorithm runs in time* $\text{poly}(|T|, k, m, b)$.

Proof. By Corollary 1, the mechanism corresponding to the reduced form that maximizes $\vec{\pi}\cdot\vec{w}$ simply maximizes virtual welfare, where the virtual values are according to $\vec{w}$. So we just need to compute the reduced form of this mechanism. What is the probability that agent i receives item j when he reports type t_i to this mechanism? If $w_{ij}(t_i)$ is negative, then this probability is clearly 0. Otherwise, it's the probability that every other agent has a lower virtual value for item j. For any $i' \neq i$, we can easily compute the probability that agent i''s virtual value for item j is less than $w_{ij}(t_i)/Pr[t_i \leftarrow \mathcal{D}_i]$: simply sum over all $t_{i'} \in T$ and check whether or not $w_{ij}(t_{i'})/Pr[t_{i'} \leftarrow \mathcal{D}_{i'}] < w_{ij}(t_i)/Pr[t_i \leftarrow \mathcal{D}_i]$. If so, add $Pr[t_{i'} \leftarrow \mathcal{D}_{i'}]$. Otherwise, add 0. Next, we can just take a product over all $i' \neq i$ of these probabilities because each agent's type is sampled independently. This will result in the probability that all other agents have a lower virtual value than agent i for item j when agent i's type is t_i. So for any i, j, t_i, we can compute $\pi_{ij}(t_i)$ corresponding to this mechanism in the desired runtime, and we can therefore compute the entire reduced form in the desired runtime as well. □

Finally, we combine Corollary 2 with the equivalence of separation and optimization (Theorem 2 in Section 2.3 of Chapter 2) and the ellipsoid algorithm (Theorem 1 in Section 2.3 of Chapter 2) to solve the linear program in Figure 3-2.

Corollary 3. *There exists a computationally efficient separation oracle, SO, for $F(\mathcal{D})$ such that* $\text{runtime}_{SO}(\ell) = \text{poly}(|T|, k, m, b, \ell)$. *Therefore, the linear program of Figure 3-2 can be solved in time* $\text{poly}(|T|, k, m, b)$. *The output will be the revenue-optimal reduced form.*

3.3 Implementation

The last step in our solution is implementing the reduced form found by solving the Linear Program of Figure 3-2. After all, recall that BMeD does not ask us to find the reduced form of a mechanism, but to find an actual mechanism: i.e. a device that can take as input a type profile $\vec{t}$ and award each item to an agent (or no one) and charge prices. To this end, we make use of Theorem 3 in Section 2.3 of Chapter 2, which states that with access to a separation oracle for a closed convex region P, any point $\vec{\pi} \in P$ can be decomposed in polynomial time into a convex combination of corners of P. In this section, we observe that this suffices to implement any feasible reduced form.

Observation 5. *Let $\vec{\pi} \in F(\mathcal{D})$ have bit complexity ℓ. Then we can decompose $\vec{\pi} = \sum_z \vec{\pi}_z q_z$ with $q_z \geq 0, \sum_z q_z = 1$ in time $\text{poly}(|T|, k, m, b, \ell)$, where each $\vec{\pi}_z$ is a corner of $F(\mathcal{D})$.*

Proof. This is a direct application of Theorem 3 from Section 2.3 in Chapter 2 and Corollary 3. □

Corollary 4. *Every feasible reduced form $\vec{\pi} \in F(\mathcal{D})$ can be implemented as a distribution over virtual-welfare maximizers. If the bit complexity of $\vec{\pi}$ is ℓ, then this distribution can be found in time $\text{poly}(|T|, k, m, b, \ell)$.*

Proof. Given any feasible reduced form $\vec{\pi}$, first write $\vec{\pi}$ as a convex combination of corners of $F(\mathcal{D})$ using the separation oracle of Corollary 3 in the decomposition algorithm of Theorem 3. By Corollary 1, $\vec{\pi}_z$ is the reduced form of the virtual welfare maximizer with virtual values according to $\vec{w}_z$. So to implement $\vec{\pi}$, simply run the mechanism corresponding to $\vec{\pi}_z$ with probability q_z. Each $\vec{\pi}_z$ is a virtual welfare maximizer. □

Proof of Theorem 4: Observation 3 reduces BMeD to developing a separation oracle for $F(\mathcal{D})$. Theorem 2 in Chapter 2 reduces obtaining a separation oracle to obtaining an algorithm that optimizes linear functions. Corollary 1 provides such an algorithm. Corollary 3 puts this all together and states that we can find the revenue-optimal reduced form in the desired runtime. Corollary 4 states that we can implement the desired reduced form in the desired runtime as well, and also provides the desired structure. ■

3.4 Conclusions

We have just shown that the revenue-optimal auction for additive agents is a distribution over virtual-welfare maximizers, and that we can find and implement this auction in polynomial time. This result also serves to prepare the reader for the complete reduction in Chapter 5. To that end, there are four important steps to remember from this approach. First, we wrote a linear program using a compact description of mechanisms (the reduced form), and showed that this linear program can be solved with black-box access to a separation oracle for a closed convex region (the space of feasible reduced forms). Second, we used the equivalence of separation and optimization to show that this separation oracle can be obtained via an optimization algorithm for the same closed convex region. Third, we showed that such an optimization algorithm simply maximizes virtual welfare on every profile, completing the "reduction." Finally, once we solve the linear program, we have to implement whatever reduced form we find as an actual mechanism, which is also done using tools from linear programming.

Chapter 4

Background: General Objectives and Non-Additive Agents

Here we introduce the required background for general objectives and non-additive agents. Section 4.1 updates notation related to mechanism design to the generalized setting, Section 4.2 provides additional preliminaries required for linear programming in our most general setting, and Section 4.3 overviews related work in algorithmic mechanism design.

4.1 Preliminaries

Throughout the preliminaries and the remainder of the thesis, we state our definitions and results when the goal is to minimize an objective (such as makespan). Everything extends to maximization objectives (such as fairness) with the obvious changes (switching $\leq$ to $\geq$, min to max, etc.). We often note the required changes. We choose to present in this way so that the reader sees an example of the framework applied to both maximization (from Chapters 2 and 3) and minimization.

Mechanism Design Setting. For the remainder of this thesis, we will be in the general setting put forth in Section 2.1. That is, the designer has to choose some possible outcome x from a set $\mathcal{F}$ of feasible outcomes, and we make no assumptions whatsoever on the structure of $\mathcal{F}$.

Agent Preferences. For the remainder of this thesis, we make no additional assumptions on the agents' preferences beyond those put forth in Section 2.1. That is, agents are still quasi-linear and risk-neutral, but no longer additive. Agents have an arbitrary valuation function $t_i : \mathcal{F} \to \mathbb{R}$. We still assume that agents are rational and therefore behave in a way that maximizes their utility.

Private Information. We still assume that each agent's type is sampled independently from a known distribution. We make no assumptions about these distributions. For instance, the value of agent i for outcomes x and x' can be correlated, but the value of agents i and i' for outcome x are independent.

Interim Descriptions. In this chapter, it will be helpful to think of the *interim description* of a mechanism. The interim allocation rule is a function that takes as input a agent i and a type $t_i \in T$ and outputs the distribution of outcomes that agent i sees when reporting type t_i, over any randomness of the mechanism and the other agents' types (assuming they report truthfully). Specifically, if the interim allocation rule of $M = (A, P)$ is X, then $X_i(t_i)$ is a distribution satisfying

$$\Pr[x \leftarrow X_i(t_i)] = \mathbb{E}_{\vec{t}_{-i} \leftarrow \mathcal{D}_{-i}} \left[\Pr[A(t_i; \vec{t}_{-i}) = x] \right],$$

where $\vec{t}_{-i}$ is the vector of types of all agents except i in $\vec{t}$, and $\mathcal{D}_{-i}$ is the distribution of $\vec{t}_{-i}$. Similarly, the *interim price rule* of the mechanism maps some agent i and type $t_i \in T$ of that agent to the expected price agent i sees when reporting t_i, i.e. $p_i(t_i) = \mathbb{E}_{\vec{t}_{-i} \leftarrow \mathcal{D}_{-i}}[P_i(t_i; \vec{t}_{-i})]$.

With interim descriptions in mind, we may rewrite the mathematical statement of Bayesian Incentive Compatible as simply $t_i(X_i(t_i)) - p_i(t_i) \geq t_i(X_i(t_i')) - p_i(t_i')$ for all i and $t_i, t_i' \in T$. We may also rewrite Individual Rationality as simply $t_i(X_i(t_i)) - p_i(t_i) \geq 0$, for all i, $t_i \in i$.

Implicit Forms. In addition to the interim description of a mechanism, it will be useful to introduce a concept that we call the *implicit form*. The implicit form takes the role of the reduced form from Chapter 2 and is used in optimization. For any feasible mechanism $M = (A, P)$ for a *BMeD*($\mathcal{F}, \mathcal{V}, \mathcal{O}$) instance, we define the three components of its implicit form $\vec{\pi}_I^M = (O^M, \vec{\pi}^M, \vec{p}^M)$ as follows.

- $O^M = \mathbb{E}_{\vec{t} \leftarrow \mathcal{D}}[O(\vec{t}, A(\vec{t}))]$. That is, O^M is the expected value of O when agents sampled from $\mathcal{D}$ play mechanism M.

- For all agents i and types $t_i, t'_i \in T$, $\pi_i^M(t_i, t'_i) = \mathbb{E}_{\vec{t}_{-i} \leftarrow \mathcal{D}_{-i}}[t_i(A(t'_i; \vec{t}_{-i}))]$. That is, $\pi_i^M(t_i, t'_i)$ is the expected value of agent i with real type t_i for reporting type t'_i to the mechanism M. The expectation is taken over any randomness in M as well as the other agents' types, assuming they are sampled from $\mathcal{D}_{-i}$.

- For all agents i and types $t_i \in T$, $p_i^M(t_i) = \mathbb{E}_{\vec{t}_{-i} \leftarrow \mathcal{D}_{-i}}[P_i(t_i; \vec{t}_{-i})]$. That is, $p_i^M(t_i)$ is the expected price paid by agent i when reporting type t_i to the mechanism M. The expectation is taken over any randomness in M as well as the other agents' types, assuming they are sampled from $\mathcal{D}_{-i}$.

We can also talk about implicit forms separately from mechanisms, and call any $(1 + k|T|^2 + k|T|)$-dimensional vector an implicit form. We say that an implicit form $\vec{\pi}_I = (O, \vec{\pi}, \vec{p})$ is *feasible* for a specific *BMeD*$(\mathcal{F}, \mathcal{V}, \mathcal{O})$ instance if there exists a feasible mechanism M for that instance such that $O \geq O^M$, $\vec{\pi} = \vec{\pi}^M$, and $\vec{p} = \vec{p}^M$. We say that the mechanism M *implements* the implicit form $\vec{\pi}_I$ when these inequalities hold. For maximization objectives, we instead constrain $O \leq O^M$.[1] We denote by $F(\mathcal{F}, \mathcal{D}, \mathcal{O})$ the set of all feasible implicit forms (with respect to a specific instance of *BMeD*$(\mathcal{F}, \mathcal{V}, \mathcal{O})$).

Goal of the designer. We consider a designer with an arbitrary objective function $\mathcal{O}$. For simplicity of exposition, we will restrict attention in this chapter to objective functions $\mathcal{O}$ that depend only on the agents' types and the outcome chosen and write the function $\mathcal{O}(\vec{t}, X)$ to emphasize this. Examples of such objectives include welfare, makespan, and fairness but not revenue. The modifications required to accommodate revenue or other objectives that depend on prices charged are straight-forward, but notationally burdensome. We often note the required modifications informally. We also remark that our results further extend to objectives that are sensitive to randomness in non-linear ways, so long as the dependence is concave in distributions over outcomes/prices for

[1]The relaxations $O \geq O^M$ for minimization, and $O \leq O^M$ for maximization objectives, instead of $O = O^M$, are required for technical reasons.

maximization objectives and convex for minimization objectives. Note that "deterministic objectives" such as makespan, fairness, welfare, and revenue that are extended to randomized outcomes by taking expectations behave linearly with respect to randomness and are therefore both concave and convex in this context. There are no modifications required to our techniques to accommodate such objectives, but formally dealing with bit complexities for such functions becomes tedious, so we omit any further discussion.

Normalization. In this chapter, we will also make use of an approximate notion of truthfulness called ϵ-Bayesian Incentive Compatible. A mechanism is ϵ-BIC if each agent can gain at most an additive ϵ by misreporting their type. That is, $t_i(X_i(t_i)) - p_i(t_i) \geq t_i(X_i(t_i')) - p_i(t_i') - \epsilon$. In order to make this additive guarantee meaningful, we assume that all types have been normalized so that $t_i(X) \in [0, 1]$ for all $X \in \mathcal{F}$. Also, because our results accommodate an additive ϵ error, we will restrict attention only to O that are poly(k)-bounded. We say that O is b-bounded if whenever $t_i(X) \in [0, 1]$ for all i, $O(\vec{t}, X) \in [0, b]$. Note that makespan and fairness are both 1-bounded, and welfare/revenue is k-bounded, so this is not a restrictive assumption.

Formal Problem Statements. We state both BMeD and GOOP for minimization objectives, but both problems are also well-defined for maximization objectives with the obvious modifications, discussed below. In the following definitions, we denote by $\mathcal{V}$ a set of potential types (i.e. functions mapping $\mathcal{F}$ to $\mathbb{R}$), and by $\mathcal{V}^\times$ the closure of $\mathcal{V}$ under addition and scalar multiplication.

BMeD($\mathcal{F}$, $\mathcal{V}$, O): Input: A finite set of types $T \subseteq \mathcal{V}$, and for each agent $i \in [k]$ a distribution $\mathcal{D}_i$ over T. Goal: Find a feasible (outputs an outcome in $\mathcal{F}$ with probability 1), BIC, and IR mechanism M that minimizes O in expectation, when k agents with types sampled from $\mathcal{D} = \times_i \mathcal{D}_i$ play M truthfully, where the minimization is with respect to all feasible, BIC, and IR mechanisms. M is said to be an α-approximation to BMeD if the expected value of O is at most α times that of the optimal mechanism.

GOOP($\mathcal{F}$, $\mathcal{V}$, O): Input: $f \in \mathcal{V}^\times$, $g_i \in \mathcal{V}$ $(1 \leq i \leq k)$, and a multiplier $w \geq 0$. Goal: find a feasible (possibly randomized) outcome $X \in \Delta(\mathcal{F})$ such that:

$$(w \cdot \mathcal{O}((g_1, \ldots, g_k), X)) + f(X) = \min_{X' \in \mathcal{F}} \{(w \cdot \mathcal{O}((g_1, \ldots, g_k), X')) + f(X')\}.$$

We define a bi-criterion approximation for GOOP. We say that X is an (α,β)-*approximation* to GOOP for some $\beta \le 1 \le \alpha$ iff:

$$\beta\,(w \cdot \mathcal{O}((g_1, \ldots, g_k), X)) + f(X) \le \alpha \left(\min_{X' \in \mathcal{F}} \{(w \cdot \mathcal{O}((g_1, \ldots, g_k), X')) + f(X')\}\right). \tag{4.1}$$

An X satisfying (4.1) with $\beta = 1$ is an α-*approximation* to GOOP, which is the familiar notion of approximation for minimization problems. If $\beta < 1$, our task becomes easier as the contribution of the $\mathcal{O}$ part to the objective we are looking to minimize is discounted.

Within the context of our reduction from BMeD to GOOP, one should interpret the function f in the GOOP instance as representing virtual welfare, and each g_i as representing the type reported by agent i.

Remark 1. *For maximization objectives $\mathcal{O}$, we replace* min *by* max *in the definition of GOOP, and we invert the direction of the inequality in* (4.1). *Moreover, the feasible range of parameters for an (α,β)-approximation are now $\alpha \le 1 \le \beta$.*

4.2 Additional Linear Programming Preliminaries

In this section, we provide additional necessary preliminaries on linear programming required for the results of the following chapter. We defer to Chapter 6 a proof of all claims, but we include definitions and statements here so that the results of the following chapter are readable.

(α,β)-approximations. Let us first define what (α,β)-approximations are for linear optimization problems. In the definition below, α and β are constants and S is a subset of the coordinates. When we write the vector $(c\vec{x}_S, \vec{x}_{-S})$, we mean the vector $\vec{x}$ where all coordinates in S have been multiplied by c.

Definition 1. *An algorithm $\mathcal{A}$ is an (α,β,S)*-minimization algorithm *for a closed convex region P iff for any input vector $\vec{w}$ the vector $\mathcal{A}(\vec{w})$ output by the algorithm satisfies:*

$$(\beta\mathcal{A}(\vec{w})_S, \mathcal{A}(\vec{w})_{-S}) \cdot \vec{w} \leq \alpha \min_{\vec{x}\in P}\{\vec{x}\cdot\vec{w}\}. \tag{4.2}$$

Given such algorithm, we also define the algorithm $\mathcal{A}_S^\beta$ that outputs $(\beta\mathcal{A}(\vec{w})_S, \mathcal{A}(\vec{w})_{-S})$.[2]

Taking $\beta = 1$ recovers the familiar notion of α-approximation, except that we do not require the output of the algorithm to lie in P in our definition. (Most meaningful applications of our framework will enforce this extra property though.) With $\beta < 1$ (respectively $\beta > 1$), the minimization (resp. maximization) becomes easier as the coordinates indexed by S are discounted (boosted) by a factor of β before comparing to $\alpha \cdot OPT$.

Weird Separation Oracles. In order to accommodate approximation algorithms in the equivalence of separation and optimization, we introduce the notion of a *weird separation oracle*. A weird separation oracle (WSO) looks kind of like a separation oracle, in that it is an algorithm that sometimes says "yes" and sometimes outputs a hyperplane. But it's behavior may be erratic, and the set of points for which the algorithm says "yes" may not be convex, closed, or even connected. We prove in Chapter 6 Theorem 5 below, which generalizes the equivalence of separation and optimization (Theorem 2 in Chapter 2) to accommodate (α,β)-approximations. Specifically, Theorem 5 states that an (α,β,S)-approximation algorithm for closed convex region P can be used to obtain a weird separation oracle for αP (by αP we mean the closed convex region P blown up by a factor of α or shrunk by a factor of α, depending on whether $\alpha \geq 1$ or $\alpha \leq 1$). Before stating the theorem formally, let's overview quickly what each property below is guaranteeing. Property 1 guarantees that WSO is consistent at least with respect to points inside αP (but may behave erratically outside of αP). Property 2 guarantees that even though the points accepted by WSO may not be in αP (or even in P), they will at least satisfy some relaxed notion of feasibility. Property 3 guarantees that WSO terminates in polynomial time. Property 4 guarantees that if we

[2]We can similarly define the concept of a (α,β,S)-*maximization algorithm* for closed convex region P by flipping the inequality in (4.2) and also switching min to max.

run Ellipsoid with WSO instead of a real separation oracle for αP, that we don't sacrifice anything in terms of optimality (although the output is only guaranteed to satisfy the notion of feasibility given in Property 2. It may be infeasible in the traditional sense, i.e. not contained in αP).

Theorem 5. *Let P be a closed convex region in $\mathbb{R}^d$ and $\mathcal{A}$ an (α,β,S)-minimization algorithm for P, for some $\alpha,\beta > 0$. Then we can design a weird separation oracle WSO for αP with the following properties:*

1. *Every halfspace output by WSO will contain αP.*
2. *Whenever $WSO(\vec{x})$ = "yes" for some input $\vec{x}$, the execution of WSO explicitly finds directions $\vec{w}_1,\dots,\vec{w}_{d+1}$ such that $\vec{x} \in Conv\{\mathcal{A}_S^\beta(\vec{w}_1),\dots,\mathcal{A}_S^\beta(\vec{w}_{d+1})\}$, and therefore $(\frac{1}{\beta}\vec{x}_S, \vec{x}_{-S}) \in Conv\{\mathcal{A}(\vec{w}_1),\dots,\mathcal{A}(\vec{w}_{d+1})\}$ as well.*
3. *Let b be the bit complexity of $\vec{x}$, ℓ an upper bound on the bit complexity of $\mathcal{A}(\vec{w})$ for all $\vec{w} \in [-1,1]^d$. Then on input $\vec{x}$, WSO terminates in time $\text{poly}(d,b,\ell,\text{runtime}_{\mathcal{A}}(\text{poly}(d,b,\ell)))$ and makes at most $\text{poly}(d,b,\ell)$ queries to $\mathcal{A}$.*
4. *Let Q be an arbitrary closed convex region in $\mathbb{R}^d$ described via some separation oracle, $\vec{c}$ a linear objective with $\vec{c}_{-S} = \vec{0}$, and $OPT = \min_{\vec{y}\in\alpha P\cap Q}\{\vec{c}\cdot\vec{y}\}$. Let also $\vec{z}$ be the output of the Ellipsoid algorithm for minimizing $\vec{c}\cdot\vec{y}$ over $\vec{y} \in \alpha P \cap Q$, but using WSO as a separation oracle for αP instead of a standard separation oracle for αP (i.e. use the exact same parameters for Ellipsoid as if WSO was a valid separation oracle for αP, and still use a standard separation oracle for Q). Then $\vec{c}\cdot\vec{z} \le OPT$, and therefore $\vec{c}\cdot(\frac{1}{\beta}\vec{z}_S, \vec{z}_{-S}) \le \frac{1}{\beta}OPT$.*

A complete proof of Theorem 5 appears in Section 6.2 of Chapter 6.

Sampling Error. Another necessary tool in the subsequent chapter will be an extension of the equivalence of separation and optimization to accommodate sampling error in the optimization algorithm. Specifically, what we will show is that if we start with a closed convex region $P \subseteq \mathbb{R}^d$ that admits a certain kind of sampling-based optimization algorithm, then there is another closed convex region P' that is nearly identical to P and is more convenient for optimization. We begin by defining these sampling-based optimization algorithms, that we call *sample-optimization* *algorithms*. In the definition below, we use the notation $\vec{w}_S$ to mean the vector $\vec{w}$ restricted to coordinates S. So by $(\vec{w}_S, 0_{-S})$ we mean the vector $\vec{w}$ where all coordinates except for S have been zeroed out.

Definition 2. *(sample-optimization algorithm) For all subsets $S \subseteq [d]$, let A_S be an exact optimization algorithm for some closed convex region $Q(S) \subseteq [0, \text{poly}(d)]^d$. We say that $\mathcal{A}$ is a sample-optimization algorithm if $\mathcal{A}(\vec{w})$ first samples a subset of coordinates $S \subseteq [d]$ according to some distribution D, then outputs $A_S((\hat{w}_S, 0_{-S}))$, where $\hat{w}$ is a vector with $\hat{w}_i = w_i/Pr[i \in S]$, and the probability in the denominator is with respect to D. We also define the sample complexity of $\mathcal{A}$, to be* $\max_i\{1/Pr[i \in S]\}$*, and the corner complexity to be* $\log N$*, where N is the smallest integer such that all corners of all $Q(S)$ are integer multiples of N. We denote by $b(\mathcal{A})$ the complexity of $\mathcal{A}$, which upper bounds the sample and corner complexity.*

Note that every exact optimization algorithm is also a sample-optimization algorithm (by sampling $S = [d]$ with probability 1).

Definition 3. *(sample-optimization region) We say that a closed convex region P is a* sample-optimization region *if there exists a sample-optimization algorithm $\mathcal{A}$, sampling S from distribution D, with the additional property that for all $\vec{w}$, there exists some $\vec{z}^* \in \text{argmax}_{\vec{x} \in P}\{\vec{x} \cdot \vec{w}\}$ such that $\mathbb{E}[\mathcal{A}(\vec{w})_i | i \in S] = z_i^*$ for all i, where again the conditioning in the expectation is taken with respect to D. We also define the complexity of P, $b(P)$ to be the minimum over all sample-optimization algorithms $\mathcal{A}$ for P of $b(\mathcal{A})$.*

For a sample-optimization region P, and corresponding sample-optimization algorithm $\mathcal{A}$ and distribution D, we also take $\text{poly}(d, b(P), 1/\epsilon)$ samples from D and denote by D' the uniform distribution over these samples. We further define the sample-optimization algorithm $\mathcal{A}'$ that first samples a subset S' of coordinates from D' and then runs $A_{S'}$. We claim that $A_{S'}$ defines a sample-optimization region P', and that with high probability over the samples taken for D', P and P' are close.

Theorem 6. *Let P be a sample-optimization region with corresponding sample-optimization algorithm $\mathcal{A}$, closed convex regions $Q(S)$, optimization algorithms A_S, and distribution D over S.*

For any ϵ, let D' be the uniform distribution over $\text{poly}(d, b(P), 1/\epsilon)$ *samples from D, and define the sample-optimization algorithm $\mathcal{A}'$ to sample S' from D' (and then run $A_{S'}$). Then $\mathcal{A}'$ defines a sample-optimization region P'. Furthermore, with probability* $1 - exp(\text{poly}(d, b(P), 1/\epsilon))$*, the following two guarantees hold:*

1. *For every $\vec{x} \in P$, there exists an $\vec{x}' \in P'$ with $|\vec{x} - \vec{x}'|_\infty \leq \epsilon$.*
2. *For every $\vec{w}$ and all $i \in [d]$, $|\mathbb{E}[\mathcal{A}(\vec{w})_i | i \in S] - \mathbb{E}[\mathcal{A}'(\vec{w})_i | i \in S]| \leq \epsilon$.*

A complete proof of Theorem 6 appears in Section 6.3 of Chapter 6.

4.3 Related Work

In this section we overview related work in algorithmic mechanism design: specifically prior work on black-box reductions, mechanisms for makespan, and mechanisms for fairness.

Black-Box Reductions in Mechanism Design. We have already established that the classical VCG mechanism can be viewed as a mechanism to algorithm reduction for the important objective of welfare [Vic61, Cla71, Gro73], and Myerson's celebrated mechanism [Mye81] as a mechanism to algorithm design reduction for revenue. Sometimes, however, these reductions ask for the design of algorithms that are computationally intractable. Work of [PSS08, BDF+10, Dob11, DV12] establishes that in prior-free settings, no approximation-preserving reduction can possibly exist even for welfare. Yet, work of [HL10, HKM11, BH11] establishes an approximation-preserving reduction in Bayesian settings for welfare. Interestingly, for the special case of revenue in single-dimensional settings, there is also a precedence of using the specific average-case structure of the problem in order to develop approximation algorithms in settings where any finite approximation seems computationally intractable at first glance [HIMM11, BHSZ13]. In the settings studied by these papers, the algorithmic problem resulting from the reduction (namely virtual welfare maximization) is highly inapproximable in the worst case. Nevertheless, they side-step this intractability by exploiting the fact that the algorithmic problem need only be solved well in an average-case sense (in particular, in expectation over the bidder's virtual values) rather than on an instance-to-instance basis, as well as the fact that, in single-dimensional settings, the virtual values have very well-understood structure. This allows for the design of polynomial-time algorithms that obtain a reasonable approximation guarantee on average, while possibly performing poorly on some instances.

Outside of welfare and revenue in single-dimensional settings, there is no further precedent of black-box reductions in mechanism design. Specifically, outside of our work there are no known black-box reductions for multi-dimensional mechanism design, and the only prior result for objectives beyond welfare or revenue is an impossibility result due to Chawla, Immorlica, and Lucier [CIL12]. Specifically, they show that there is no black-box reduction from truthfully minimizing makespan to algorithmically minimizing makespan even in single-dimensional settings. On this front, our work provides the first positive result for black-box reductions beyond welfare or revenue in single-dimensional settings, and greatly improves the state of the art.

Makespan. A long line of work following the seminal paper of Nisan and Ronen [NR99] addresses the question of "how much better can the optimal makespan be when compared to the optimal makespan obtained by a truthful mechanism?" The same paper showed that the answer is at most a factor of k, and also that the answer is at least 2 for deterministic, DSIC, prior-free mechanisms. It was later shown that the answer is at least $1 + \phi$ (the golden ratio) as $k \to \infty$ [CKV07, KV07], and that the answer is in fact k for the restricted class of anonymous mechanisms [ADL12]. It is conjectured that the answer is indeed k for all deterministic, DSIC, prior-free mechanisms. Similar (but slightly different) bounds are known for the same question with respect to randomized prior-free mechanisms [NR99, MS07, CKK07, LY08a, LY08b, Lu09]. More recently, the same question has been studied for prior-independent (rather than prior-free) mechanisms [CHMS13]. Prior-independent mechanisms make distributional assumptions about the processing times, but do not use the specifics of the distributions, just their properties. In particular, when the processing times are drawn from a machine-symmetric product distribution with Monotone Hazard Rate (MHR) marginals, Chawla, Hartline, Malec and Sivan show that the answer is at most a factor of $O(m/k)$, and at most a factor of $O(\sqrt{\log k})$, when all processing times are i.i.d. [CHMS13]. Without the MHR assumption, they obtain bicriterion results.[3] The question has also been studied in *related machines* settings, where each job has a size (public) and each machine has a speed (private). Due to the single-dimensional nature of the problem, the answer is now exactly a factor of 1 [AT01]. Thus, focus has shifted towards the same question for computationally efficient truthful mechanisms, and constant-factor approximations [AT01] and PTAS's [DDDR08, CK10] are known.

The focus of our work is different than most previous works, in that we do not study the gap between the algorithmic and the mechanismic optimum. Instead, our focus is computational, aiming for (approximately) optimal and computationally efficient mechanisms, regardless of how their performance compares to the performance of optimal algorithms. Still, prior work has already made some progress on this problem: we know that the VCG mechanism is a computationally efficient k-approximation [NR99], and that the mechanisms of [CHMS13] provide an approximation ratio of $O(m/k)$ when the prior is a machine-symmetric product distribution with MHR marginals, and a ratio of $O(\sqrt{\log k})$ if additionally the marginals are i.i.d. In addition, the NP-hardness result of [LST87] implies that our problem is also NP-hard to approximate better than $3/2 - \epsilon$, for any $\epsilon > 0$. On this front, our work greatly improves the state-of-the-art as we give the first constant-factor approximations for unrestricted settings. (The guarantees of [CHMS13] are constant in settings where $m = O(k)$ or $k = O(1)$ and the prior is a machine-symmetric product distribution with MHR marginals.) Indeed, our approximation guarantee (factor of 2) matches that of the best polynomial-time algorithm for makespan minimization [LST87].

Max-Min Fairness. Fair division has been studied extensively in Mathematics and Economics for over 60 years; see e.g. [Ste48, Kna46, BT96]. Several different flavors of the problem have been studied: divisible or indivisible goods (in our terminology: jobs), with or without monetary transfers to the players (in our terminology: machines), and several different notions of fairness. For the allocation of indivisible goods, virtually all mechanisms proposed in the literature do not optimize max-min fairness, aiming instead at other fairness guarantees (such as envy-freeness or

[3]Specifically, they obtain the same bounds with respect to a different benchmark, namely the optimal expected makespan using only a fraction of the k machines.

proportionality), very commonly trade off value from received items by the players with monetary transfers to or from the players in the fairness guarantee, and are often susceptible to strategic manipulations. For max-min fairness, Bezakova and Dani [BD05] propose a prior-free mechanism for 2 players, which guarantees half of the optimal fairness, albeit under restrictions on the strategies that the players can use. They also show that max-min fairness cannot be optimally implemented truthfully in prior-free settings. In fact, Mu'alem and Schapira show that no DSIC, deterministic, prior-free mechanism can obtain any approximation to the optimal max-min fairness [MS07].

Perhaps the poor state-of-the-art on mechanisms for max-min fairness owes to the fact that, already as an algorithmic problem (i.e. even when the players' true values for the items are assumed exactly known), max-min fairness has proven quite challenging, indeed significantly more so than makespan minimization. In particular, all state-of-the-art algorithms only provide polynomial approximation guarantees [BD05, AS07, KP07, BCG09, CCK09], while the best known computational hardness result is just a factor of 2 [BD05]. Specifically, the guarantees lying on the Pareto boundary of what is achievable in polynomial time for the unrestricted problem are approximation factors of $m - k + 1$ ([BD05]), $\tilde{O}(\sqrt{k})$ ([AS07]), and $O(m^{1/\epsilon})$ ([BCG09, CCK09]). Due to this, a *restricted* version of the problem is often studied, where every job has a fixed processing time p_j, and every machine is either capable or incapable of processing each job (that is, $p_{ij} \in \{p_j, 0\}$). For the restricted version, the state of the art is an $O(\log\log k/\log\log\log k)$-approximation due to Bansal and Sviridenko [BS06]. Asadpour, Feige, and Saberi [AFS08] also proved that the integrality gap of the configuration LP used in [BS06] has an integrality gap of 5, and provided a heuristic (but not polynomial-time) rounding algorithm. $O(1)$-approximations were obtained by Bateni, Charikar and Guruswami and independently by Chakrabarty, Chuzhoy and Khanna by further restricting the graph structure of which machines can process which jobs (i.e. by limiting the number of machines that can process each specific job or requiring that this graph be acyclic) [BCG09, CCK09].

In view of this literature, our results provide the first approximately optimal mechanisms for max-min fairness. Indeed, our approximation factors match those of approximation algorithms on the Pareto boundary of what is achievable in polynomial time [BD05, AS07]. So, in particular, our mechanisms cannot be strictly improved without progress on the algorithmic front. Obtaining these mechanisms is already quite involved (see Chapter 7), and we leave open for future investigation the problem of matching the bounds obtained in [BCG09, CCK09] for the general problem, [AFS08, BS06] for the restricted problem, and [BCG09, CCK09] for the futher restricted version.

In summary, our work greatly improves the state of the art for black-box reductions in mechanism design, and furthermore our new framework yields the first poly-time mechanisms for two paradigmatic algorithmic problems (namely, makespan and fairness).

Chapter 5

The Complete Reduction

In this section we provide our black-box reduction from mechanism to algorithm design (formally, BMeD to GOOP) in its complete generality. The proof of correctness follows a similar skeleton to that of Theorem 4 from Chapter 3, but several extra algorithmic steps are required due to the increased generality. Most of these algorithmic steps are taken separately in Chapter 6, as they are of independent interest.

Our work in this chapter provides meaningful structure on the optimal mechanism for any objective in addition to the computational implications. Specifically, we show that the optimal mechanism takes the following form:

1. Each agent reports their type.
2. These types are transformed into virtual types. This transformation is randomized and pre-computed by a linear program.
3. The outcome maximizing O+virtual welfare is selected.
4. Payments are charged to ensure that the mechanism is truthful. These payments are also pre-computed by a linear program.

In the following theorem statement (and throughout the remainder of this chapter), we let b denote an upper bound on the bit complexity of $O(\vec{t}, x)$, $Pr[t_i \leftarrow \mathcal{D}_i]$, and $t_i(x)$ over all $x \in \mathcal{F}$, $\vec{t_i} \in T$, $\vec{t} \in \times_{i\in[k]} T$.

Theorem 7. *Let G be an (α,β)-approximation algorithm for GOOP($\mathcal{F}$,$\mathcal{V}$,$\mathcal{O}$), for some $\alpha \geq 1 \geq \beta > 0$, and some minimization objective $\mathcal{O}$. Also, fix any $\epsilon > 0$. Then there is an approximation algorithm for BMeD($\mathcal{F}$,$\mathcal{V}$,$\mathcal{O}$) that makes* $\text{poly}(|T|, b, k, 1/\epsilon)$ *calls to G, and runs in time* $\text{poly}(|T|, b, k, 1/\epsilon, \text{runtime}_G(\text{poly}(|T|, b, k, 1/\epsilon)))$. *If OPT is the optimal obtainable expected value of $\mathcal{O}$ for some BMeD instance, then the mechanism M output by the algorithm on that instance yields $\mathbb{E}[\mathcal{O}(M)] \leq \frac{\alpha}{\beta}OPT + \epsilon$, and is ϵ-BIC. These guarantees hold with probability at least* $1-\exp(\text{poly}(|T|, b, k, 1/\epsilon))$. *Furthermore, the output mechanism is feasible and can be implemented in time* $\text{poly}(|T|, b, k, 1/\epsilon, \text{runtime}_G(\text{poly}(|T|, b, k, 1/\epsilon)))$. *These guarantees hold with probability* 1.

Sections 5.1, 5.2, and 5.3 below provide a complete proof of Theorem 7. As Theorem 7 is quite general and a bit abstract, we provide in Section 5.4 several instantiations of Theorem 7 to important or previously studied settings.

5.1 A Linear Program

We begin by stating the linear program used in our algorithm in Figure 5-1. Note the similarity to the linear program of Figure 3-2 used in Chapter 3, the only difference is that we've replaced the reduced form with the implicit form. Before continuing with our analysis, we should address why this replacement is necessary. The issue with using the reduced form in settings with non-additive agents is that there is simply not enough information in the reduced form to determine if a mechanism is truthful or not. The following example illustrates why.

Example. Let there be a single agent and two items. Her value for receiving both items together is 2, and her value for receiving a single item (or nothing) is 0. Consider the mechanism M that awards both items with probability $1/2$ and charges a price of 1, and the mechanism M' that awards a single item chosen uniformly at random and charges a price of 1. Then the reduced forms of M and M' are identical: each awards item 1 with probability $1/2$, item 2 with probability $1/2$, and charges a price of 1. But the agent's expected value for participating in M is 1, whereas her expected value for participating in M' is 0. Therefore, M is individually rational but M' is not.

Variables:

- $\pi_i(t_i, t_i')$, for all agents i and types $t_i, t_i' \in T$, denoting the expected value obtained by agent i when their true type is t_i but they report t_i' instead.
- $p_i(t_i)$, for all agents i and types $t_i \in T$, denoting the expected price paid by agent i when they report type t_i.
- O, denoting the expected value of $\mathcal{O}$.

Constraints:

- $\pi_i(t_i, t_i) - p_i(t_i) \geq \pi_i(t_i, t_i') - p_i(t_i')$, for all agents i, and types $t_i, t_i' \in T$, guaranteeing that the implicit form $(O, \vec{\pi}, \vec{p})$ is BIC.
- $\pi_i(t_i, t_i) - p_i(t_i) \geq 0$, for all agents i, and types $t_i \in T$, guaranteeing that the implicit form $(O, \vec{\pi}, \vec{p})$ is individually rational.
- $(O, \vec{\pi})$ is feasible. **A separation oracle is needed for this.**

Minimizing:

- O, the expected value of $\mathcal{O}$ when played truthfully by agents sampled from $\mathcal{D}$.

Figure 5-1: A linear programming formulation for BMeD.

Observation 6. *The linear program of Figure 5-1 finds the implicit form of a solution to a given BMeD instance. Furthermore, is SO is a separation oracle for the space of feasible implicit forms, this LP can be solved in time* $\text{poly}(k, |T|, b, \text{runtime}_{SO}(k, |T|, b))$.

Remark 2. *To optimize revenue instead of $\mathcal{O}$, one can modify the linear program of Figure 5-1 by removing entirely the variable O. Feasibility now only constrains $\vec{\pi}$ instead of $(O, \vec{\pi})$, and the objective function should be the expected revenue, $\sum_i Pr[t_i \leftarrow \mathcal{D}_i] \cdot p_i(t_i)$ instead of O (and should be maximized instead of minimized).*

Remark 3. *To optimize an objective $\mathcal{O}$ that depends on $\vec{t}$, X, and $\vec{p}$, modify the feasibility constraint so that feasibility constraints all of $(O, \vec{\pi}, \vec{p})$ (instead of just $(O, \vec{\pi})$).*

Unfortunately, even with our approximation-preserving equivalence of separation and optimization, we can't always get a meaningful separation oracle for the space of feasible implicit forms, which we denote by $F(\mathcal{F}, \mathcal{D}, \mathcal{O})$. To cope with this, we further develop the equivalence of separation and optimization framework to accommodate an additive sampling error. Essentially, we replace the closed convex region $F(\mathcal{F}, \mathcal{D}, \mathcal{O})$ with the closed convex region $F(\mathcal{F}, \mathcal{D}', \mathcal{O})$, where $\mathcal{D}'$ is a uniform distribution over polynomially many samples from $\mathcal{D}$. We postpone until Section 6.3 a proof that this procedure is valid, but rewrite here the linear program we actually solve in Figure 5-2. We also note that this replacement is the reason that ϵ appears in Theorem 7.

Variables:

- $\pi_i(t_i, t_i')$, for all agents i and types $t_i, t_i' \in T$, denoting the expected value obtained by agent i when their true type is t_i but they report t_i' instead.
- $p_i(t)$, for all agents i and types $t_i \in T$, denoting the expected price paid by agent i when they report type t_i.
- O, denoting the expected value of $\mathcal{O}$.

Constraints:

- $\pi_i(t_i, t_i) - p_i(t_i) \geq \pi_i(t_i, t_i') - p_i(t_i')$, for all agents i, and types $t_i, t_i' \in T$, guaranteeing that the implicit form $(O, \vec{\pi}, \vec{p})$ is BIC.
- $\pi_i(t_i, t_i) - p_i(t_i) \geq 0$, for all agents i, and types $t_i \in T$, guaranteeing that the implicit form $(O, \vec{\pi}, \vec{p})$ is individually rational.
- $(O, \vec{\pi}) \in F(\mathcal{F}, \mathcal{D}', \mathcal{O})$. **A separation oracle is needed for this.**

Minimizing:

- O, the expected value of $\mathcal{O}$ when played truthfully by agents sampled from $\mathcal{D}$.

Figure 5-2: A linear program for BMeD, replacing $F(\mathcal{F}, \mathcal{D}, \mathcal{O})$ with $F(\mathcal{F}, \mathcal{D}', \mathcal{O})$.

5.2 The Space of Feasible Implicit Forms

Like Chapter 3, our approach to solve the LP in Figure 5-2 by obtaining a separation oracle for the space $F(\mathcal{F}, \mathcal{D}', \mathcal{O})$, making use of the equivalence of separation and optimization. However, in order to accommodate approximation, we need to develop some new machinery. As this machinery is of independent interest, we treat it separately in Chapter 6.

We now begin by proving that the linear program of Figure 5-2 can be solved computationally efficiently by making use of machinery from Chapter 6 (specifically, Theorem 5).

Proposition 2. *With black-box access to an $(\alpha,\beta,\{O\})$-optimization algorithm, $\mathcal{A}$, for $F(\mathcal{F},\mathcal{D}',O)$, one can use the weird separation oracle guaranteed by Theorem 5 inside the ellipsoid algorithm to solve the linear program of Figure 5-2. The Ellipsoid algorithm will terminate in time* $\text{poly}(|T|, b, k, 1/\epsilon, \text{runtime}_{\mathcal{A}}(\text{poly}(|T|, b, k, 1/\epsilon)))$*, and will output a truthful implicit form $\vec{\pi}_I$. The objective component O of $\vec{\pi}_I$ will satisfy $O \leq \alpha OPT'$, where OPT' is the value of O in the optimal solution to the LP of Figure 5-2. Furthermore, the algorithm will explicitly output a list of $d+1 = \text{poly}(|T|, k)$ directions $\vec{w}_1, \ldots, \vec{w}_{d+1}$ such that $\vec{\pi}_I \in Conv\{\mathcal{A}^{\beta}_{\{O\}}(\vec{w}_1), \ldots, \mathcal{A}^{\beta}_{\{O\}}(\vec{w}_{d+1})\}$.*

Proof. Theorem 5 guarantees that the linear program can be solved in the desired runtime, and that the desired directions $\vec{w}_1, \ldots, \vec{w}_{d+1}$ will be output. It is clear that any implicit form satisfying the constraints is truthful.

Let now OPT'_α denote the value of the LP in Figure 5-2 using a real separation oracle for $\alpha F(\mathcal{F},\mathcal{D}',O)$. Theorem 5 also guarantees that $O \leq OPT'_\alpha$. So we just need to show that $OPT'_\alpha \leq \alpha OPT'$.

To see this, first observe that the origin satisfies every constraint in the linear program not due to $F(\mathcal{F},\mathcal{D}',O)$ (i.e. the truthfulness constraints) with equality. Therefore, if any implicit form $\vec{\pi}_I$ is truthful, so is the implicit form $\alpha\vec{\pi}_I$. This immediately implies that $OPT'_\alpha = \alpha OPT'$. □

Next, we show that one can obtain an $(\alpha,\beta,\{O\})$-optimization algorithm for $F(\mathcal{F},\mathcal{D}',O)$ given black-box access to an (α,β)-approximation algorithm for $GOOP(\mathcal{F},\mathcal{V},O)$. We first need a technical lemma giving context to dot products in the space of feasible implicit forms.

Proposition 3. *Let $\vec{w}$ be a direction in $[-1,1]^{1+k|T|^2+k|T|}$. Define the virtual type of agent i with type t' as $\hat{t}'_{\vec{w}} = \sum_{t\in T} \frac{w_i(t,t')}{\Pr[t'\leftarrow\mathcal{D}_i]} t(\cdot)$. That is, $\hat{t}'_{\vec{w}}(X) = \sum_{t\in T} \frac{w_i(t,t')}{\Pr[t'\leftarrow\mathcal{D}_i]} t(X)$. Define also the virtual objective $O'_{\vec{w}}$ as:*

$$O'_{\vec{w}}(\vec{t}\,', X) = w_O \cdot O(\vec{t}\,', X) + \sum_i \hat{t}'_{i\vec{w}}(X)$$

Then for any mechanism $M = (A, P)$, if $\vec{\pi}_I^M$ denotes the implicit form of M with respect to $\mathcal{D}$, $\vec{\pi}_I^M \cdot \vec{w}$ is exactly the expected virtual objective of M on type profiles sampled from $\mathcal{D}$. Formally:

$$\vec{\pi}_I^M \cdot \vec{w} = \mathbb{E}_{\vec{t}\,' \leftarrow \mathcal{D}}[O'_{\vec{w}}(\vec{t}\,', A(\vec{t}\,'))]$$

Proof. We begin by expanding the dot product:

$$\vec{\pi}_I^M \cdot \vec{w} = O \cdot w_O + \sum_i \sum_{t,t' \in T} \pi_i(t,t') w_i(t,t')$$

Recalling that $\pi_i(t,t')$ is the value of t for the interim allocation seen by t', we rewrite $\pi_i(t,t') = t(X_i(t'))$, yielding:

$$\vec{\pi}_I^M \cdot \vec{w} = O \cdot w_O + \sum_i \sum_{t,t' \in T} w_i(t,t') \cdot t(X_i(t'))$$

Now we multiply and divide each term by $Pr[t' \leftarrow \mathcal{D}_i]$, and rearrange the sum to sum first over i, then t', then t:

$$\vec{\pi}_I^M \cdot \vec{w} = O \cdot w_O + \sum_i \sum_{t' \in T} Pr[t' \leftarrow \mathcal{D}_i] \sum_{t \in T} \frac{w_i(t,t')}{Pr[t' \leftarrow \mathcal{D}_i]} \cdot t(X_i(t'))$$

Now let's interpret the sum above: O is exactly the expected value of $\mathcal{O}$, taken across all profiles, so $O \cdot w_O$ is just the expected value multiplied by w_O. $X_i(t')$ is the interim allocation seen by agent i with type t', and this allocation is being evaluated by the function $\sum_{t \in T} \frac{w_i(t,t')}{Pr[t' \leftarrow \mathcal{D}_i]} t(\cdot)$, which is exactly $\hat{t}'_{\vec{w}}(\cdot)$. As this is the virtual type of agent i with type t' evaluating the interim allocation seen, this is exactly the virtual value of type t'. So summing over all agents and all types, $Pr[t' \leftarrow \mathcal{D}_i]$ times the virtual value of t' yields exactly the expected virtual welfare. So $\vec{\pi}_I^M$ is exactly the expected value of the objective plus the expected virtual welfare, which is exactly the expected virtual objective. □

With Proposition 3, we can now show how to get an $(\alpha, \beta, \{O\})$-optimization algorithm for $F(\mathcal{F}, \mathcal{D}', \mathcal{O})$ using an (α, β)-approximation algorithm for GOOP.

Proposition 4. *Let all types in the support of $\mathcal{D}$ (and therefore $\mathcal{D}'$ as well) be in the set $\mathcal{V}$. Then with black-box access to G, an (α,β)-approximation algorithm for GOOP($\mathcal{F},\mathcal{V},\mathcal{O}$), one can obtain an $(\alpha,\beta,\{O\})$-optimization algorithm, $\mathcal{A}$, for $F(\mathcal{F},\mathcal{D}',\mathcal{O})$. The algorithm terminates in time* $\text{poly}(|T|, b, k, 1/\epsilon, \text{runtime}_G(\text{poly}(|T|, b, k, 1/\epsilon)))$. *Furthermore, given as input any direction $\vec{w}$, one can implement in time* $\text{poly}(|T|, b, k, 1/\epsilon, \text{runtime}_G(\text{poly}(b, k, 1/\epsilon)))$ *a feasible mechanism M whose implicit form with respect to $\mathcal{D}'$, $\vec{\pi}_0^M$, satisfies $\vec{\pi}_0^M = \mathcal{A}(\vec{w})$.*

Proof. Let's first consider the case that $w_O \geq 0$. The $w_O < 0$ case will be handled with one technical modification. Consider first that for any fixed $\vec{t}'$, the problem of finding X that minimizes $O'_{\vec{w}}(\vec{t}', X)$ is an instance of *GOOP*($\mathcal{F},\mathcal{V},\mathcal{O}$). Simply let $w = w_O$, $f = \sum_i \sum_{t \in T} \frac{w_i(t,t')}{\Pr[t' \leftarrow \mathcal{D}_i]} \cdot t(\cdot)$, and $g_i = t'_i(\cdot)$.

So with black-box access to an (α,β)-approximation algorithm, G, for *GOOP*($\mathcal{F},\mathcal{V},\mathcal{O}$), let $M = (A, P)$ be the mechanism that on profile $\vec{t}'$ runs G on input $w = w_O$, $f = \sum_i \sum_{t \in T} \frac{w_i(t,t')}{\Pr[t' \leftarrow \mathcal{D}_i]} \cdot t(\cdot)$, $\vec{g} = \vec{t}'$. We therefore get that the mechanism M satisfies the following inequality:

$$\mathbb{E}_{\vec{t}' \leftarrow \mathcal{D}'}[\beta \cdot w_O \cdot O(\vec{t}', A(\vec{t}')) + \sum_i \sum_{t \in T} \frac{w_i(t,t')}{\Pr[t' \leftarrow \mathcal{D}_i]} \cdot t(A(\vec{t}'))]$$

$$\leq \alpha \mathbb{E}_{\vec{t}' \leftarrow \mathcal{D}'}[\min_{X' \in \mathcal{F}} \{w_O \cdot O(\vec{t}', X') + \sum_i \sum_{t \in T} \frac{w_i(t,t')}{\Pr[t' \leftarrow \mathcal{D}_i]} \cdot t(X')\}]$$

By Proposition 3, this then implies that:

$$(\beta O^M, \vec{\pi}^M, \vec{p}^M) \cdot \vec{w} \leq \alpha \min_{\vec{x} \in F(\mathcal{F},\mathcal{D}',\mathcal{O})} \{\vec{x} \cdot \vec{w}\}$$

This exactly states that $\vec{\pi}_0^M$ is an (α,β,O)-approximation. It is also clear that we can compute $\vec{\pi}_0^M$ efficiently: $\mathcal{D}'$ has polynomially many profiles in its support, so we can just run $\mathcal{A}$ on every profile and see what it outputs, then take an expectation to compute the necessary quantities of the implicit form. Note that this computation is the reason we bother using $\mathcal{D}'$ at all, as we cannot compute these expectations exactly in polynomial time for $\mathcal{D}$ as the support is exponential.

Now we state the technical modification to accommodate $w_O < 0$. Recall that for any feasible implicit form $(O, \vec{\pi}, \vec{p})$, that the implicit form $(O', \vec{\pi}, \vec{p})$ is also feasible for any $O' \geq O$. So if $w_O < 0$, simply find any feasible implicit form, then set the O component to ∞. This yields a feasible implicit form with $\vec{\pi}_I \cdot \vec{w} = -\infty$, which is clearly an (α, β, O)-approximation (in fact, it is a $(1, 1, O)$-approximation). If instead the problem has a maximization objective, we may w.l.o.g. set $O = 0$ in the implicit form we output, which means that the contribution of O is completely ignored. So we can use the exact same approach as the $w_O \geq 0$ case and just set $w_O = 0$.

So let $\mathcal{A}$ be the algorithm that runs G on every profile as described, and computes the implicit form of this mechanism with respect to $\mathcal{D}'$. $\mathcal{A}$ clearly terminates in the desired runtime. Finally, to implement a mechanism whose implicit form $\vec{\pi}_0^M$ matches $\mathcal{A}(\vec{w})$, simply run G with the required parameters on every profile.

$\square$

Taken together, the above propositions provide an algorithm to find an implicit form $\vec{\pi}_I$ whose objective component is within an α-factor of optimal (for the linear program of Figure 5-2, at least). The only remaining step is to implement it. As mentioned earlier, there's a bit of a catch here because $\vec{\pi}_I$ may not even be feasible. We instead approximately implement $\vec{\pi}_I$, losing a factor of β in the objective component but leaving the others untouched.

Proposition 5. *Let $\vec{\pi}_I = (O, \vec{\pi}, \vec{p})$ be the implicit form and $\vec{w}_1, \ldots, \vec{w}_{d+1}$ the auxiliary information output by the algorithm of Proposition 2, when using some (α,β)-approximation algorithm G for GOOP($\mathcal{F}, \mathcal{V}, \mathcal{O}$) in order to get the required $(\alpha, \beta, \{O\})$-optimization algorithm for $F(\mathcal{F}, \mathcal{D}', \mathcal{O})$, via Proposition 4. Then one can implement in time* $\text{poly}(d, \text{runtime}_G(\text{poly}(b, k, 1/\epsilon)))$ *a mechanism M whose implicit form $\vec{\pi}'^M_I$ with respect to $\mathcal{D}'$ satisfies:*

- $O'^M = O/\beta$.
- $\vec{\pi}'^M = \vec{\pi}$.
- $\vec{p}'^M = \vec{p}$.

Proof. By Proposition 2, the implicit form $\vec{\pi}'_I$ output by the linear program of Figure 5-2 is in the convex hull of $\{\mathcal{A}_S^\beta(\vec{w}_1), \ldots, \mathcal{A}_S^\beta(\vec{w}_{d+1})\}$. Therefore, the implicit form $\vec{\pi}'_I = (O/\beta, \vec{\pi}, \vec{p})$ is in the convex hull of $\{\mathcal{A}(\vec{w}_1), \ldots, \mathcal{A}(\vec{w}_{d+1})\}$. Therefore, we can implement $\vec{\pi}'_I$ with respect to $\mathcal{D}'$ by randomly sampling a direction $\vec{w}_j$ according to the convex combination, and then implementing the corresponding $\mathcal{A}(\vec{w}_j)$. Call this mechanism M. By Proposition 4, this can be done time polynomial in the desired quantities. □

Now, we can combine Propositions 2 through 5 to prove the following corollary, which is essentially Theorem 7 minus the issue of $\mathcal{D}$ vs. $\mathcal{D}'$.

Corollary 5. *Let G be an* (α,β)*-approximation algorithm for GOOP(*$\mathcal{F}$,$\mathcal{V}$,O*), for some* $\alpha \geq 1 \geq \beta > 0$*, and some minimization objective* O*. Then one can find a mechanism M in time* $\text{poly}(|T|, b, k, 1/\epsilon, \text{runtime}_G(\text{poly}(|T|, b, k, 1/\epsilon)))$ *with the following guarantees. The expected value of the objective with respect to* $\mathcal{D}'$*,* O'^M *will satisfy* $O'^M \leq \alpha OPT'/\beta$*, where* OPT' *is the value of O in the optimal solution to the LP of Figure 5-2. Furthermore, M is feasible, and can be implemented in time* $\text{poly}(|T|, b, k, 1/\epsilon, \text{runtime}_G(\text{poly}(|T|, b, k, 1/\epsilon)))$*. The implicit form of M with respect to* $\mathcal{D}'$ *is truthful.*

To get from Corollary 5 to Theorem 7, the last step is to quantify how much is lost by using $\mathcal{D}'$ instead of $\mathcal{D}$ in our linear program. We conclude this section with two remarks on how to modify this approach to optimize revenue or other objectives that depend on the prices charged. Again, the modifications required are straight-forward, but notationally burdensome.

Remark 4. *When optimizing revenue instead of* O*, the O term may be dropped everywhere starting from Proposition 3, and also in the definition of GOOP. In other words, determining whether or not an imlicit form* $\vec{\pi}$ *is feasible again requires black-box access to an algorithm optimizing virtual welfare.*

Remark 5. *When optimizing an objective* O *that depends on* $\vec{t}$*, X, and* $\vec{p}$*, we must add a* $\vec{p}$ *term everywhere starting from Proposition 3, and also to the definition of GOOP. In other words, in addition to the virtual objective, there will be some linear function of* $\vec{p}$ *appearing in the statement of GOOP that can be interpreted as a "virtual revenue" term. In this setting, the goal of GOOP is to find an allocation and charge prices optimizing the virtual objective plus virtual revenue.*

5.3 Consequences of using $\mathcal{D}'$

Here, we quantify the loss incurred by replacing $\mathcal{D}$ with $\mathcal{D}'$. Essentially, our goal is to show that $F(\mathcal{F}, \mathcal{D}, \mathcal{O})$ satisfies the hypotheses put forth in Section 4.2 of Chapter 4, and that $F(\mathcal{F}, \mathcal{D}', \mathcal{O})$ is the resulting closed convex region from applying our techniques. We first show that $F(\mathcal{F}, \mathcal{D}, \mathcal{O})$ is a sample-optimization region.

Proposition 6. *$F(\mathcal{F}, \mathcal{D}, \mathcal{O})$ is a sample-optimization region.*

Proof. We first define the closed convex regions $Q(S) \subseteq \mathbb{R}^{1+k|T|^2+k|T|}$. Define $S_{\vec{t}\,'}$ to be a set containing $1 + k|T|$ coordinates. First, the coordinate corresponding to the objective is in $S_{\vec{t}\,'}$ for all $\vec{t}\,'$. Next, for each agent i, the $|T|$ coordinates that correspond to agent i's reported type being t'_i are in $S_{\vec{t}\,'}$ as well. To make notation cleaner, we simply refer to the closed convex region $Q(S_{\vec{t}\,'})$ as $Q(\vec{t}\,')$ instead. Now, let $\vec{y} \in Q(\vec{t}\,')$ if and only if:

- $y_i(t, t_0) = 0$, for all i, t and $t_0 \neq t'_i$.
- There exists an $X \in \Delta(\mathcal{F})$ such that $\mathcal{O}(\vec{t}\,', X) \leq y_O$ and $t(X) = y_i(t, t'_i)$ for all i and $t \in T$.

We first show that each $Q(\vec{t}\,')$ is convex, and observe that we have already characterized an optimization algorithm for each $Q(\vec{t}\,')$.

Observation 7. *$Q(\vec{t}\,')$ is convex for all $\vec{t}\,'$.*

Proof. Let $\vec{y}, \vec{y}\,' \in Q(\vec{t}\,')$. Then clearly $c\vec{y} + (1-c)\vec{y}\,'$ has $y_i(t, t_0) = 0$ for all $t_0 \neq t'_i$. Furthermore, there are some distributions X and X' witnessing the second condition for $\vec{y}$ and $\vec{y}\,'$ respectively, so $cX + (1-c)X'$ witnesses the second condition for $c\vec{y} + (1-c)\vec{y}\,'$. □

Observation 8. *In the closed convex region $Q(\vec{t}\,')$, the algorithm, $A_{\vec{t}\,'}$ that optimizes in direction $\hat{w}$ finds the outcome $x \in \mathcal{F}$ minimizing $\hat{w}_O \mathcal{O}(\vec{t}\,', x) + \sum_i \sum_{t \in T} \hat{w}_i(t, t'_i) \cdot t(x)$ and outputs the vector $\vec{y}^x$ satisfying:*

1. *$y_i(t, t_0) = 0$ for all i, t and $t_0 \neq t'_i$.*
2. *$y_i(t, t'_i) = t(x)$ for all i, t.*

3. $y_O = O(\vec{t}', x)$, *if* $\hat{w}_O \geq 0$, *or* $y_O = \infty$ *if* $\hat{w}_O < 0$.

Proof. The proof is immediate by the definition of $Q(\vec{t}')$. We must have $y_i(t, t_0) = 0$ for all $t_0 \neq t'_i$. Also, for whatever point $\vec{y}$ is chosen, there is some $X \in \Delta(\mathcal{F})$ witnessing $\vec{y} \in Q(\vec{t}')$. As $\vec{y}$ is a corner w.l.o.g., X is in fact a point mass selecting some $x \in \mathcal{F}$ with probability 1. Therefore, we must have $y_i(t, t'_i) = t(x)$. Lastly, note we may set any value of $y_O \geq O(\vec{t}', x)$ we choose. We want y_O to be as small as possible if $w_O \geq 0$, or as large as possible otherwise. □

Notice now that the algorithm in Observation 8 looks familiar: it is exactly finding the outcome $x \in \mathcal{F}$ that optimizes a virtual objective. With this, we're ready to complete our proof and show that $F(\mathcal{F}, \mathcal{D}, \mathcal{O})$ is a sample-optimization region.

Define a sample-optimization algorithm $\mathcal{A}$ to sample $\vec{t}' \leftarrow \mathcal{D}$, and select the subset of coordinates $S_{\vec{t}'}$. Observe now that for all coordinates (i, t, t'), $Pr[(i, t, t') \in S_{\vec{t}'}] = Pr[t' \leftarrow \mathcal{D}_i]$. We also know from Proposition 3 that the mechanism optimizing in direction $\vec{w}$ within $F(\mathcal{F}, \mathcal{D}, \mathcal{O})$ optimizes the virtual objective on every profile. By Observation 8, the optimization algorithm for $Q(\vec{t}')$ is exactly optimizing the virtual objective on profile $\vec{t}'$. If M is the mechanism optimizing the virtual objective corresponding to $\vec{w}$ on every profile, it is also easy to see that $\pi_i^M(t, t') = \mathbb{E}[(A_{\vec{t}'}((\hat{w}_{\vec{t}'}, 0_{-\vec{t}'})))_i(t, t')]$ for all i, t, t' and that $\pi_O^M = \mathbb{E}[(A_{\vec{t}'}((\hat{w}_{\vec{t}'}, 0_{-\vec{t}'})))_O]$. □

Now that we know that $F(\mathcal{F}, \mathcal{D}, \mathcal{O})$ is a sample-optimization region, we'd like to make use of Theorem 6. If $\mathcal{D}'$ is a uniform distribution over samples from $\mathcal{D}$, then it's easy to see that taking $P = F(\mathcal{F}, \mathcal{D}, \mathcal{O})$ as in the statement of Theorem 6 results in $P' = F(\mathcal{F}, \mathcal{D}', \mathcal{O})$. So we just need to figure out how many samples we need to take for $\mathcal{D}'$ in order for Theorem 6 to take effect. We first remark that we can do some inconsequential rounding on the input probabilities and values to BMeD which will aid in this.

Remark 6. *We may without loss of generality assume the following about the input to a BMeD instance without affecting the objective or truthfulness by more than an additive ϵ (and hence these losses can be absorbed into the ϵ in the statement of Theorem 7):*

1. *All probabilities $Pr[t \leftarrow \mathcal{D}_i]$ are at least $\epsilon/\text{poly}(k, |T|)$. To see this, observe that we can modify any mechanism to first with probability $\epsilon/\text{poly}(k, |T|)$ ignore the type reported by*

agent i and select for her a uniform random type instead. This modification does not affect the truthfulness for agent i at all, but may affect truthfulness for other agents up to an additive $\epsilon/\text{poly}(k, |T|)$*, and may affect the objective by up to an additive* $\epsilon/\text{poly}(k, |T|)$*. We can then view this new mechanism as just the original mechanism where the input comes from different distributions with* $Pr[t \leftarrow \mathcal{D}_i] \geq \epsilon/\text{poly}(k, |T|)$*. So consider any optimal mechanism* M' *for the rounded probability distributions. The reasoning above shows that the quality of* M' *is at most* $OPT + \epsilon k/\text{poly}(k, |T|)$*. Furthermore, we have also argued that we can implement any mechanism for the rounded distributions when agents are instead drawn from the original distributions by resampling their type uniformly at random with probabilty* $\epsilon/\text{poly}(k, |T|)$ *losing only an additional* $\epsilon/\text{poly}(k, |T|)$ *in truthfulness.*

2. *All values* $t_i(x)$ *are integer multiples of* $\epsilon/\text{poly}(k)$*. To see this, simply round every* $t_i(x)$ *down to the nearest multiple of* $\epsilon/\text{poly}(k)$ *and treat this as the input instead. This can only affect the truthfulness for agent i by up to an additive* $\epsilon/\text{poly}(k)$*, and the objective by up to an additive* $\epsilon/\text{poly}(k)$ *and has no affect on the truthfulness of other agents. By the same reasoning as the previous bullet, this means that the optimal mechanism* M' *for the rounded values has quality at most* $OPT + \epsilon k/\text{poly}(k)$*. Furthermore, if* M'' *is the mechanism that first rounds the reported types, and then runs* M'*, then it's clear that the truthfulness of* M'' *is only* $\epsilon k/\text{poly}(k)$ *worse than the truthfulness of* M'*. And it's also clear that the quality of* M'' *is at most* $\epsilon k/\text{poly}(k)$ *worse than* M'*. This last statement holds whether or not* M' *is optimal.*

3. *Both bullet points above suggest that we may first round the input distributions and values as described, find a mechanism* M'*, and then apply some "unrounding" procedure to obtain a mechanism* M'' *for the original input. Both bullet points have argued that if* M' *was approximately optimal for the rounded input, then it is also approximately optimal for the original input, up to an additional additive* $\epsilon/\text{poly}(k, |T|)$*. Furthermore, the total loss in both truthfulness and quality from the procedure is bounded by* $\epsilon/\text{poly}(k, |T|)$*, and therefore can be absorbed into the* ϵ *in the statement of Theorem 7.*

With Remark 6 in mind, we can state formally the conclusion of this section below. Combined with Corollary 5, this completes the proof of Theorem 7.

Proposition 7. *Taking* poly$(1/\epsilon, k, |T|)$ *samples for* $\mathcal{D}'$ *yields the following with probability* $1 - exp(\text{poly}(|T|, b, k, 1/\epsilon))$*:*

1. *For all* $\vec{\pi}_I^M \in F(\mathcal{F}, \mathcal{D}, O)$*, there exists a* $\vec{\pi'}_I^M \in F(\mathcal{F}, \mathcal{D}, O)$ *such that* $|\vec{\pi}_I^M - \vec{\pi'}_I^M|_\infty \leq \epsilon$*.*

2. *Every mechanism M that optimizes a virtual objective on every profile has an implicit form with respect to* $\mathcal{D}$*,* $\vec{\pi}_I^M$*, and with respect to* $\mathcal{D}'$*,* $\vec{\pi'}_I^M$ *satisfying* $|\vec{\pi'}_I^M - \vec{\pi'}_I^M|_\infty \leq \epsilon$*.*

Proof. Both claims are immediate corollaries of Theorem 6 in Chapter 4 as long as we check that we've taken enough samples. With Remark 6, all values of all agents for all outcomes are integer multiples of $\epsilon/\text{poly}(k, |T|)$. As all the coordinates of all corners of each $Q(\vec{t}\,')$ represent the value of an agent for some outcome, this means that the corner complexity of each $Q(\vec{t}\,')$ is $\log(\text{poly}(k, |T|)/\epsilon)$. Also by Remark 6, we may take the minimum probability $\Pr[t' \leftarrow \mathcal{D}_i]$ to be $\epsilon/\text{poly}(k, |T|)$. This means that the sample complexity of $F(\mathcal{F}, \mathcal{D}, O)$ is bounded above by $\text{poly}(k, |T|)/\epsilon$. Therefore, $b(F(\mathcal{F}, \mathcal{D}, O)) = \text{poly}(k, |T|, 1/\epsilon)$. Plugging back into Theorem 6, this means that forming $\mathcal{D}'$ with $\text{poly}(k, |T|, 1/\epsilon)$ samples from $\mathcal{D}$ suffices. □

Proof of Theorem 7: With Proposition 7, it's clear that $OPT' \leq OPT + \epsilon$. Combining this with Corollary 5 proves the approximate optimality of the mechanism output. Also, as the implicit form $\vec{\pi'}_I^M$ is BIC, Proposition 7 immediately guarantees that the implicit form $\vec{\pi}_I^M$ is ϵ-BIC. The remaining guarantees (that hold with probability 1) have already been shown in Corollary 5. ■

5.4 Instantiations of Theorem 7

In this section, we provide several instantiations of Theorem 7.

Revenue - Matching Markets. The designer has a single copy of each of m heterogeneous items for sale to k unit-demand agents, and the goal is to maximize expected revenue. In this setting, GOOP is simply asking for a max-weight matching in a bipartite graph with k agents on the left and m items on the right. As GOOP can be solved exactly in poly-time, we obtain the following:

Theorem 8. *There is a PTAS for revenue maximization in matching markets. For any desired $\epsilon > 0$, the output mechanism is ϵ-BIC, and has expected revenue at least $OPT - \epsilon$ with probability at least $1 - \exp(\text{poly}(b, |T|, k, m, 1/\epsilon))$. Furthermore, the output mechanism is feasible, and can be implemented in time $\text{poly}(b, |T|, k, m, 1/\epsilon)$ with probability 1. The runtime of the algorithm is $\text{poly}(b, |T|, k, m, 1/\epsilon)$. Furthermore, the allocation rule of the mechanism (randomly) assigns a virtual value to each type of each agent for each item, and on every profile selects the matching that maximizes virtual welfare.*

Revenue - Arbitrary Feasibility Constraints. The designer has multiple copies of each of m heterogenous goods for sale to k additive agents, and the goal is to maximize expected revenue. Furthermore, there are feasibility constraints $\mathcal{F}$ on which agents may simultaneously receive which items. Formally, one may interpret $\mathcal{F} \subseteq 2^{[k]\times[m]}$, and the set $\{(i_1, j_1), \ldots, (i_n, j_n)\} \in \mathcal{I}$ as referencing the fact that it is feasible to simultaneously award item j_ℓ to agent i_ℓ for all ℓ. Note that this generalizes matching markets where $\mathcal{F}$ is the set of all matchings. In this setting, GOOP asks for the maximum weight set $S \in \mathcal{F}$, where the weight of element (i, j) is the virtual value of agent i for item j. Referring to this problem as $MaxWeight(\mathcal{F})$, Theorem 7 implies the following:

Theorem 9. *Let G be a poly-time α-approximation algorithm for $MaxWeight(\mathcal{F})$. Then there is an α-approximation algorithm for revenue maximization subject to feasibility constraints $\mathcal{F}$ (formally,* BMeD*($\mathcal{F}$, {additive functions with coefficients $t_{ij} \in [0, 1]$}, revenue)). For any desired $\epsilon > 0$, the output mechanism is ϵ-BIC, and has expected revenue at least $\alpha OPT - \epsilon$ with probability at least $1 - \exp(\text{poly}(b, |T|, k, m, 1/\epsilon))$. Furthermore, the output mechanism is feasible, and can be implemented in time $\text{poly}(b, |T|, k, m, 1/\epsilon)$ with probability 1. The runtime of the algorithm is $\text{poly}(b, |T|, k, m, 1/\epsilon)$. Furthermore, the allocation rule of the mechanism (randomly) assigns a virtual value to each type of each agent for each item, and on every profile runs G on the corresponding virtual values.*

A specific instantiation of Theorem 9 might have $\mathcal{F}$ be the intersection of p matroids, in which case we would take $\alpha = 1/p$, or $\alpha = 1$ for the case of $p = 2$.

Job Scheduling on Unrelated Machines. The designer has m jobs that he wishes to schedule on k machines. The processing time of job j on machine i is t_{ij}, and the goal is to minimize the expected makespan. In this setting, GOOP is asking for a solution to the problem of makespan minimization with costs. This problem is NP-hard to approximate within any finite factor, but a poly-time $(1, 1/2)$-approximation exists and is developed in Chapter 7. Making use of this, we obtain the following theorem.

Theorem 10. *There is a polynomial-time 2-approximation algorithm for truthful job scheduling on unrelated machines (formally the problem* $\text{BMeD}(\{0,1\}^{km}$*, {additive functions with coefficients* $t_{ij} \in [0,1]$*}, Makespan)). For any desired* $\epsilon > 0$*, the output mechanism is* ϵ*-BIC, and has expected makespan at most* $2OPT + \epsilon$ *with probability at least* $1 - \exp(\text{poly}(b, k, m, 1/\epsilon))$*. Furthermore, the output mechanism is feasible, and can be implemented in time* $\text{poly}(b, k, m, 1/\epsilon)$ *with probability* 1*. The runtime of the algorithm is* $\text{poly}(b, k, m, 1/\epsilon)$*.*

Fair Allocation of Indivisible Goods. The designer has m gifts that he wishes to award to k children. Each child i has a value of t_{ij} for item j, and the goal is to maximizes the expected fairness. In this setting, GOOP is asking for a solution to the problem of fairness maximization with costs. This problem is NP-hard to approximate within any finite factor, but a poly-time $(1, \min\{\tilde{O}(\sqrt{k}), m-k+1\})$-approximation exists and is developed in Chapter 7. Making use of this, we obtain the following theorem.

Theorem 11. *There is a polynomial-time* $\min\{\tilde{O}(\sqrt{k}), m - k + 1\}$*-approximation algorithm for truthful fair allocation of indivisible goods (formally the problem* $\text{BMeD}(\{0,1\}^{km}$*, {additive functions with coefficients* $t_{ij} \in [0,1]$*}, Fairness)). For any desired* $\epsilon > 0$*, the output mechanism is* ϵ*-BIC, and has expected fairness at least* $\min\{\tilde{O}(\sqrt{k}), m - k + 1\}OPT - \epsilon$ *with probability at least* $1 - \exp(\text{poly}(b, k, m, 1/\epsilon))$*. Furthermore, the output mechanism is feasible, and can be implemented in time* $\text{poly}(b, k, m, 1/\epsilon)$ *with probability* 1*. The runtime of the algorithm is* $\text{poly}(b, k, m, 1/\epsilon)$*.*

Chapter 6

Equivalence of Separation and Optimization

In this section we detail a general tool in linear programming, colloquially called "The Equivalence of Separation and Optimization." A well-known result of Khachiyan (Theorem 1 in Section 2.3 of Chapter 2) states that, given black-box access to a separation oracle for any d-dimensional closed convex region P, linear functions can be optimized over P in time polynomial in d and the runtime of the separation oracle. A lesser-known result of Grötschel, Lovász, and Schrijver and independently Karp and Papadimitriou [GLS81, KP80] states that the converse holds as well: given black-box access to an optimizer of linear functions over any d-dimensional closed convex region P, one can obtain a separation oracle for P that runs in time polynomial in d and the runtime of the optimizer. In other words, the problems of separation and optimization are computationally equivalent.

While Khachiyan's result has obvious implications for solving linear programs, it's less clear that the converse has a practical use. After all, the most common usage of separation oracles is as a means to optimize linear functions, so why would we ever want to obtain a separation oracle if we already had an algorithm for optimization? One compelling use of the converse is to optimize linear functions over the *intersection* of two convex regions, P and Q. The ability to optimize linear functions over P and Q separately tells us absolutely nothing about how to optimize over

$P \cap Q$, but it does allow us to get a separation oracle for both P and Q (via the converse). This then immediately gives us a separation oracle for $P \cap Q$ (simply run the separation oracle for P followed by the separation oracle for Q if P accepts), and Khachiyan's ellipsoid algorithm then allows us to optimize over $P \cap Q$. There are indeed other uses of separation oracles beyond optimization, including the decomposition algorithm outlined in Theorem 3 of Section 2.3 in Chapter 2, the first poly-time algorithm for submodular minimization [GLS81], and numerous others [GLS81, KP80].

In Section 6.1 we provide a very similar proof of the exact equivalence to the one given in [GLS81] in order to acclimate the reader with these techniques. In Sections 6.2 we extend the equivalence to be robust to traditional and bicriterion approximations and in 6.3 we extend the equivalence to be robust to sampling error.

6.1 Exact Optimization

Here we provide a proof of the original equivalence of separation and optimization (stated as Theorem 2 in Section 2.3 of Chapter 2) so that the reader is familiar with these techniques. Our extensions will follow a similar approach but require more technical tools.

Proof of Theorem 2: Because P is a closed convex region, we know that $\vec{x} \in P$ if and only if $\vec{x} \cdot \vec{w} \leq \max_{\vec{y} \in P}\{\vec{y} \cdot \vec{w}\}$ for all directions $\vec{w} \in \mathbb{R}^d$. In fact, the same is true even if we consider only normalized $\vec{w} \in [-1, 1]^d$. So we can try to write a program to search over all $\vec{w} \in \mathbb{R}^d$ for one that is violated by $\vec{x}$ as in Figure 6-1:

Lemma 2. *The linear program of Figure 6-1 has value* > 0 *if and only if* $\vec{x} \notin P$.

Proof. First, assume that $\vec{x} \notin P$. Then there is some direction $\vec{w} \in [-1, 1]^d$ such that $\vec{x} \cdot \vec{w} > \max_{\vec{y} \in P}\{\vec{y} \cdot \vec{w}\}$. Then consider setting $t = \max_{\vec{y} \in P}\{\vec{y} \cdot \vec{w}\}$, and the point $(\vec{w}, t)$. It is clear that this point is feasible, and also that $\vec{x} \cdot \vec{w} > t$. So $(\vec{w}, t)$ bears witness that the value of the linear program is > 0.

Now, assume that the value of the linear program is > 0, and consider the point $\vec{w}, t$ witnessing this. Then we clearly have $\vec{x} \cdot \vec{w} > t$, as the value of the LP is > 0. And because $(\vec{w}, t)$ is feasible,

Variables:

- $\vec{w}$, a d-dimensional direction.
- t, a helper variable used to bound $\max_{\vec{y}\in P}\{\vec{y}\cdot\vec{w}\}$.

Constraints:

- $\vec{w}\in[-1,1]^d$, guaranteeing that the entries of $\vec{w}$ have been normalized.
- $t\geq\vec{y}\cdot\vec{w}\ \forall\vec{y}\in P$, guaranteeing that t correctly bounds $\max_{\vec{y}\in P}\{\vec{y}\cdot\vec{w}\}$.

Maximizing:

- $\vec{x}\cdot\vec{w}-t$, the gap between $\vec{x}\cdot\vec{w}$ and $\max_{\vec{y}\in P}\{\vec{y}\cdot\vec{w}\}$.

Figure 6-1: A linear programming formulation to find a violated hyperplane.

we must also have $t\geq\max_{\vec{y}\in P}\{\vec{y}\cdot\vec{w}\}$. Putting these together, we have found a direction $\vec{w}$ such that $\vec{x}\cdot\vec{w}>\max_{\vec{y}\in P}\{\vec{y}\cdot\vec{w}\}$, and therefore $\vec{x}\notin P$. □

In light of Lemma 2, we just need to solve the linear program of Figure 6-1 efficiently. Unfortunately, there are infinitely many constraints of the form $t\geq\vec{y}\cdot\vec{w}$, so we can't hope to list them all explicitly. Instead, we will define an efficient separation oracle for these constraints, $\widehat{SO}$. Observe that if *any* constraint of the form $t\geq\vec{y}\cdot\vec{w}$ is violated for some $\vec{y}\in P$, then certainly $t\geq\text{argmax}_{\vec{y}\in P}\{\vec{y}\cdot\vec{w}\}$ is also violated. So rather than check each constraint, $\widehat{SO}$ simply checks this single constraint. And finally, we can actually find $\text{argmax}_{\vec{y}\in P}\{\vec{y}\cdot\vec{w}\}$ by simply running the optimization algorithm $\mathcal{A}$ in direction $\vec{w}$. Figure 6-2 details explicitly this approach.

Putting everything back together, the complete approach is the following. First, solve the LP in Figure 6-2. By the reasoning above, this is also a solution to the LP in Figure 6-1. Finally, by Lemma 2, if $(\vec{w}^*,t^*)$ denotes the solution output, we either have $\vec{x}\cdot\vec{w}^*\leq t^*$, and $\vec{x}\in P$, or $\vec{x}\cdot\vec{w}^*>t^*$, but $\vec{y}\cdot\vec{w}^*\leq t$ for all $\vec{y}\in P$, and therefore $(\vec{w}^*,t)$ is a violated hyperplane.

■

Variables:

- $\vec{w}$, a d-dimensional direction.
- t, a helper variable used to bound $\max_{\vec{y}\in P}\{\vec{y}\cdot\vec{w}\}$.

Constraints:

- $\vec{w} \in [-1, 1]^d$, guaranteeing that the entries of $\vec{w}$ have been normalized.
- $\widehat{SO}(\vec{w}, t) =$
  - "yes" if $t \geq \mathcal{A}(\vec{w}) \cdot \vec{w}$;
  - the violated hyperplane $t' \geq \mathcal{A}(\vec{w}) \cdot \vec{w}'$ otherwise.

Maximizing:

- $\vec{x}\cdot\vec{w} - t$, the gap between $\vec{x}\cdot\vec{w}$ and $\max_{\vec{y}\in P}\{\vec{y}\cdot\vec{w}\}$.

Figure 6-2: A reformulation of Figure 6-1 using a separation oracle.

6.2 Multiplicative Approximations

In this section we detail how to extend the equivalence of separation and optimization to accommodate a multiplicative bi-criterion approximation error that we call (α,β)-approximations (defined in Section 4.2 of Chapter 4). We remind the reader that this section's main theorem (Theorem 5 in Section 4.2 of Chapter 4) states that an (α,β,S)-approximation algorithm for closed convex region P can be used to obtain a weird separation oracle (WSO) for αP. We call the separation oracle weird because it behaves kind of like a separation oracle in that in sometimes says "yes" and sometimes outputs hyperplanes, but the set of points accepted by the separation oracle isn't necessarily closed, convex or even connected.

We prove Theorem 5 for minimization algorithms, noting that the proof for maximization algorithms is nearly identical after switching $\leq$ for $\geq$ and min for max where appropriate. The proof is a series of five lemmas, one for each property, plus a technical lemma (Lemma 6). We begin by defining our weird separation oracle in Figure 6-3. Throughout this section, b denotes an upper bound on the bit complexity of $\vec{y}$, and ℓ an upper bound on the bit complexity of $\mathcal{A}_S^\beta(\vec{w})$ for all $\vec{w}$.

Lemma 3. *If $\mathcal{A}_S^\beta$ is an (α,β,S)-minimization algorithm for the closed convex region P, then every*

$WSO(\vec{y}) =$

- "**Yes**" if the ellipsoid algorithm with N iterations ($N = \text{poly}(d, b, \ell)$) outputs "infeasible" on the following problem:

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

  - $\vec{w} \in [-1, 1]^d$;
  - $t \geq \vec{y} \cdot \vec{w} + \delta$ ($\delta = 2^{-\text{poly}(d,b,\ell)}$);
  - $\widehat{WSO}(\vec{w}, t) =$
    - "yes" if $t \leq \mathcal{A}_S^{\beta}(\vec{w}) \cdot \vec{w}$;[a]
    - the violated hyperplane $t' \leq \mathcal{A}_S^{\beta}(\vec{w}) \cdot \vec{w}'$ otherwise.

- If a feasible point $(t^*, \vec{w}^*)$ is found, output the violated hyperplane $\vec{w}^* \cdot \vec{x} \leq t^*$.

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

Figure 6-3: A weird separation oracle.

halfspace output by WSO contains αP.

Proof. If WSO outputs a halfspace $(\vec{w}^*, t^*)$, then we must have $\widehat{WSO}(\vec{w}^*, t^*) = \text{"yes"}$, implying that $t^* \leq \mathcal{A}_S^{\beta}(\vec{w}^*) \cdot \vec{w}^*$. Because $\mathcal{A}$ is an (α, β, S)-approximation, we know that $\mathcal{A}_S^{\beta}(\vec{w}^*) \cdot \vec{w}^* \leq \alpha \min_{\vec{x} \in P}\{\vec{x} \cdot \vec{w}^*\} = \min_{\vec{x} \in \alpha P}\{\vec{x} \cdot \vec{w}^*\}$. Therefore, every $\vec{x} \in \alpha P$ satisfies $t^* \leq \vec{w}^* \cdot \vec{x}$, and the halfspace contains αP. □

Lemma 4. *On input $\vec{y}$, WSO terminates in time* $\text{poly}(d, b, \ell, \text{runtime}_{\mathcal{A}}(\text{poly}(d, b, \ell)))$ *and makes at most* $\text{poly}(d, b, \ell)$ *queries to $\mathcal{A}$.*

Proof. By Theorem 1, it is clear that $\widehat{WSO}$ is queried at most $\text{poly}(N, d, b, \ell, BC(\delta))$ times, where $BC(\delta)$ is the bit complexity of δ. Each execution of $\widehat{WSO}$ makes one call to $\mathcal{A}$. So the total runtime is clearly as desired. □

Lemma 5. *Let Q be an arbitrary convex region in $\mathbb{R}^d$ described via some separation oracle, and $OPT = \min_{\vec{y} \in \alpha P \cap Q}\{\vec{c} \cdot \vec{y}\}$ for some vector $\vec{c}$. Let also $\vec{z}$ be the output of the Ellipsoid algorithm for minimizing $\vec{c} \cdot \vec{y}$ over $\vec{y} \in \alpha P \cap Q$, but using WSO as a separation oracle for αP instead of a standard separation oracle (i.e. use the exact parameters for the Ellipsoid algorithm as if WSO*

was a valid separation oracle for αP, and still use a standard separation oracle for Q). Then $\vec{z} \cdot \vec{c} \leq OPT$.

Proof. When the Ellipsoid algorithm tries to minimize $\vec{c} \cdot \vec{x}$, it does a binary search over possible values C, and checks whether or not there is a point $\vec{x}$ satisfying $\vec{x} \cdot \vec{c} \leq C$, $\vec{x} \in Q$, and $WSO(\vec{x}) =$ "yes". If there is a point $\vec{x}$ satisfying $\vec{x} \cdot \vec{c} \leq C$, $\vec{x} \in Q$, and $\vec{x} \in \alpha P$, then clearly every halfspace output by the separation oracle for Q contains $\vec{x}$, and so does the halfspace $\vec{y} \cdot \vec{c} \leq C$. Furthermore, by Lemma 3, every halfspace output by WSO contains $\vec{x}$ as well. Therefore, if $OPT \leq C$, the Ellipsoid algorithm using WSO will find a feasible point and continue its binary search. Therefore, the algorithm must conclude with a point $\vec{z}$ satisfying $\vec{z} \cdot \vec{c} \leq OPT$. □

Lemma 6. *Let W be any set of directions $\vec{w}'$. Then the following closed convex region ($P(W)$) is either empty, or has volume at least $2^{-\text{poly}(d,b,\ell)}$:*

$$t - \vec{x} \cdot \vec{w} \geq \delta$$

$$t \leq \mathcal{A}_S^{\beta}(\vec{w}') \cdot \vec{w}, \ \forall \vec{w}' \in W$$

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

Proof. First, it will be convenient to add the vacuous constraint $t \leq d$ to the definition of the closed convex region. It is vacuous because it is implied by the existing constraints, but useful for analysis. Define $P'(W)$ by removing the first constraint. That is, $P'(W)$ is the intersection of the following halfspaces:

$$t \leq d$$

$$t \leq \mathcal{A}_S^{\beta}(\vec{w}') \cdot \vec{w}, \ \forall \vec{w}' \in W$$

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

If there is a point in $P(W)$, then there is clearly a corner $(t^*, \vec{w}^*)$ of $P'(W)$ in $P(W)$ as well. As all constraints in $P'(W)$ have bit complexity b, we must have $t^* - \vec{x} \cdot \vec{w}^* \geq \gamma$, with $\gamma = 2^{-\text{poly}(d,b,\ell)}$. Let $\delta = \gamma/4$. Therefore, the point $(t^*/2, \vec{w}^*/2)$ is also clearly in $P(W)$, and satisfies $t^* - \vec{x} \cdot \vec{w}^* \geq 2\delta$.

Now, consider the box $B = (\times_{i=1}^{d}[\frac{w_i^*}{2}, \frac{w_i^*}{2} + \frac{\delta}{2d}]) \times [\frac{t^*}{2} - \delta, \frac{t^*}{2} - \frac{\delta}{2}]$. We claim that we have $B \subseteq P(W)$ as well. Let $(t, \vec{w})$ denote an arbitrary point in B. It is clear that we have $\vec{w} \in [-1, 1]^d$ still, as we had $\vec{w}^*/2 \in [-1/2, 1/2]^d$ to start with. As each coordinate of $\vec{x}$ and $\mathcal{A}_S^{\beta}(\vec{w}')$ for all $\vec{w}'$ is in $[-1, 1]$, it is easy to see that:

$$(\vec{w}^*/2) \cdot \vec{x} - \frac{\delta}{2} \le \vec{w} \cdot \vec{x} \le (\vec{w}^*/2) \cdot \vec{x} + \frac{\delta}{2}$$

$$(\vec{w}^*/2) \cdot \mathcal{A}_S^{\beta}(\vec{w}') - \frac{\delta}{2} \le \vec{w} \cdot \mathcal{A}_S^{\beta}(\vec{w}') \le (\vec{w}^*/2) \cdot \mathcal{A}_S^{\beta}(\vec{w}') + \frac{\delta}{2} \ \forall \vec{w}' \in W$$

As we must have $t \le \frac{t^*}{2} - \frac{\delta}{2}$, and we started with $t^* \le \vec{w}^* \cdot \mathcal{A}_S^{\beta}(\vec{w}')$ for all $\vec{w}' \in W$, it is clear that we still have $t \le \vec{w} \cdot \mathcal{A}_S^{\beta}(\vec{w}')$ for all $\vec{w}' \in W$. Finally, since we started with $t^*/2 - \vec{x} \cdot \vec{w}^*/2 \ge 2\delta$, and $t \ge t^*/2 - \delta$, we still have $t - \vec{x} \cdot \vec{w} \ge \delta$.

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

Lemma 7. *Whenever $WSO(\vec{y})$ = "yes" for some input $\vec{y}$, the execution of WSO explicitly finds directions $\vec{w}_1, \dots, \vec{w}_{d+1}$ such that $\vec{y} \in Conv\{\mathcal{A}_S^{\beta}(\vec{w}_1), \dots, \mathcal{A}_S^{\beta}(\vec{w}_{d+1})\}$.*

Proof. Consider the intersection of halfspaces $P(W)$ as in the statement of Lemma 6.

If $\vec{y} \notin Conv(\{\mathcal{A}_S^{\beta}(\vec{w}) | \vec{w} \in W\})$, there exists some weight vector $\vec{w}^* \in [-1, 1]^d$ such that $\vec{y} \cdot \vec{w}^* < \min_{\vec{w} \in W}\{\mathcal{A}_S^{\beta}(\vec{w}) \cdot \vec{w}^*\}$. And for appropriately chosen $\delta = 2^{-\text{poly}(d,b,\ell)}$, we also have $\vec{y} \cdot \vec{w}^* \le \min_{\vec{w} \in W}\{\mathcal{A}_S^{\beta}(\vec{w}) \cdot \vec{w}^*\} - \delta$.

So if $\vec{y} \notin Conv(\{\mathcal{A}_S^{\beta}(\vec{w}) | \vec{w} \in W\})$, consider the point $\vec{w}^*, t^*$, with $t^* = \min_{\vec{w} \in W}\{\mathcal{A}_S^{\beta}(\vec{w}) \cdot \vec{w}^*\}$. By the reasoning in the previous paragraph, it's clear that $(\vec{w}^*, t^*)$ is in $P(W)$.

So consider an execution of WSO that accepts the point $\vec{y}$. Then taking W to be the set of directions $\vec{w}$ queried by the Ellipsoid algorithm during the execution of WSO, $P(W)$ is exactly the closed convex region of halfspaces that the Ellipsoid algorithm maintained. Because we did N iterations of the algorithm, and the volume of the maintained closed convex region decreases by a multiplicative factor of $1 - 1/\text{poly}(d)$ at each iteration, taking N to be a large enough $\text{poly}(d, b, \ell)$ implies that the volume of $P(W)$ is at most $2^{-\text{poly}(d,b,\ell)}$ for any desired polynomial. So taking N sufficiently large would mean that the volume of $P(W)$ is in fact smaller than the $2^{-\text{poly}(d,b,\ell)}$ in the statement of Lemma 6, and therefore $P(W)$ must be empty. This necessarily means that $\vec{y} \in Conv(\{\mathcal{A}_S^\beta(\vec{w})|\vec{w} \in W\})$, as otherwise the previous paragraphs prove that the above intersection of halfspaces wouldn't be empty.

So we may take $\vec{w}_1, \ldots, \vec{w}_{d+1}$ to be (an appropriately chosen subset of) the directions queried by the Ellipsoid algorithm during the execution of WSO and complete the proof of the lemma. □

Proof of Theorem 5: Combine Lemmas 3, 4, 5, and 7. ■

6.3 Sampling Approximation

In this section we show how to extend the equivalence of separation and optimization to accommodate sampling error in an optimization algorithm over certain kinds of closed convex regions. Our approach is quite different from the previous section in that we directly show that these certain kinds of closed convex regions can be approximated well by simpler closed convex regions that are easier to optimize over. Recall that definitions of sample-optimization algorithms and sample-optimization regions were given in Section 4.2 of Chapter 4.

For a given sample-optimization region P and sample-optimization algorithm $\mathcal{A}$, denote as in Section 4.2 of Chapter 4 by D the distribution from which S is sampled. If we could just enumerate the support of D, then we could compute exactly $\mathbb{E}[\mathcal{A}(\vec{w})_i|i \in S]$ for all i, and this would give us z_i^*. So we can use $\mathcal{A}$ to obtain an exact optimization algorithm for P, but in potentially exponential time. A natural approach is to just take polynomially many samples from D to get an additive approximation, but unfortunately the equivalence of separation and optimization framework cannot accommodate additive approximation error unless it is exponentially small (smaller than what can be guaranteed by sampling). Instead, we take polynomially many samples from D once and for all, let D' be the uniform distribution over these samples, and define a new closed convex region using D' and $\mathcal{A}$.

We can now consider instead the sample-optimization algorithm $\mathcal{A}'$ that first samples a subset S' from D' instead of D. Because we can explicitly enumerate the support of D', we can compute exactly the expected value $\mathbb{E}[\mathcal{A}'(\vec{w})_i|i \in S']$. But we need to guarantee that this is close to the original expectation, *for all* $\vec{w}$. Henceforth, we overload notation and refer to the algorithm that runs $\mathcal{A}'(\vec{w})$, then enumerates the support of D' and computes exactly the expected value $\mathbb{E}[\mathcal{A}'(\vec{w})_i | i \in S']$ for all coordinates as $\mathcal{A}'$ instead.

We show that $\mathcal{A}'$ is a useful algorithm by claiming that in fact, $\mathcal{A}'$ is an exact optimization algorithm for a different closed convex region P'. Therefore, we can use the equivalence of separation and optimization with respect to the closed convex region P' with no issues. After this, the remaining task is showing that P and P' are similar closed convex regions with high probability. Without this, being able to optimize and separate over P' is useless. Section 6.3.1 shows that $\mathcal{A}'$ does indeed define a closed convex region P'. Sections 6.3.2 and 6.3.3 show that P and P' are similar regions.

6.3.1 $\mathcal{A}'$ Defines a Closed Convex Region

In this section, we show that $\mathcal{A}'$ is an exact optimization algorithm for some closed convex region P', which we describe now. Let $\{\vec{y}(X)\}_{X \subseteq [d]}$ be an collection of vectors with each $\vec{y}(X) \in Q(X)$. $Q(X)$ are the closed convex regions used in the definition of a sample-optimization algorithm. Let then $\vec{y}$ be the vector with $y_i = \sum_X Pr[S' = X | i \in S'] y(X)_i$. Define P' to be the set of all $\vec{y}$ that can be obtained this way. We begin with a quick observation that gives context to why we define P' this way:

Observation 9. $\mathcal{A}'(\vec{w})_i = \sum_X Pr[S' = X | i \in S'] A_X((\hat{w}_X, 0_{-X}))$.

Proof. The right hand side exactly computes the expected value of $\mathcal{A}'(\vec{w})_i$ conditioned on $i \in S'$. □

To give context to P', we are essentially defining the space of points resulting from replacing $A_X((\hat{w}_X, 0_{-X}))$ with any point in $Q(X)$. We now prove that P' is a closed convex region.

Observation 10. *P' is closed and convex.*

Proof. Take any two points $\vec{y}, \vec{z} \in P'$. Then there exist some $\vec{y}(X), \vec{z}(X) \in Q(X)$ such that $y_i = \sum_X Pr[S' = X | i \in S'] y(X)_i$ and $z_i = \sum_X Pr[S' = X | i \in S'] z(X)_i$. As $Q(X)$ is convex, we also have $cy(X) + (1 - x)z(X) \in Q(X)$, and such a point clearly satisfies

$$cy_i + (1-c)z_i = \sum_X Pr[S' = X | i \in S'] cz(X)_i + (1-c)y(X)_i.$$

So P' is convex. To see that P' is closed, observe that any convergent sequence of points in P' corresponds to convergent sequences of collections of points in $Q(X)$. As each $Q(X)$ is closed, the limit points are contained in each $Q(X)$, and therefore the limit point of the original convergent sequence is also contained in P'. □

We now prove a useful lemma about computing inner products in P'.

Lemma 8. *Let $\vec{y} \in P'$ with $y_i = \sum_X Pr[S' = X | i \in S'] y(X)_i$. Then $\vec{y} \cdot \vec{w} = \sum_X Pr[S' = X] \vec{y}(X) \cdot (\hat{w}_X, 0_{-X})$.*

Proof. We can write $\vec{y} \cdot \vec{w} = \sum_i y_i w_i$. We can then rewrite this as:

$$\sum_i y_i w_i = \sum_i \sum_X Pr[S' = X | i \in S'] y(X)_i w_i$$

Using Bayes' rule and the fact that $Pr[S' = X \wedge i \in S'] = Pr[S' = X]$ for all $X \ni i$ we get:

$$\sum_i \sum_X Pr[S' = X | i \in S'] y(X)_i w_i = \sum_i \sum_X Pr[S' = X] y(X)_i \frac{w_i}{Pr[i \in S']}$$

Observing that $\hat{w}_i = w_i / Pr[i \in S']$ and rearraging the sum to sum first over X and then over i we get:

$$\sum_i \sum_X Pr[S' = X] y(X)_i \frac{w_i}{Pr[i \in S']} = \sum_X Pr[S' = X] \sum_{i \in X} y(X)_i \hat{w}_i = \sum_X Pr[S' = X] \vec{y}(X) \cdot (\hat{w}_X, 0_{-X})$$

□

With Lemma 8, we can now prove the main result of this section. Lemma 8 will, however, still be useful in the following sections.

Proposition 8. $\mathcal{A}'(\vec{w}) \in \text{argmax}_{\vec{x} \in P'} \{\vec{x} \cdot \vec{w}\}$ *for all* $\vec{w} \in \mathbb{R}^d$.

Proof. By definition, every $\vec{y} \in P'$ has some corresponding $\vec{y}(X) \in Q(X)$ with $y_i = \sum_X Pr[S' = X|i \in S']y(X)_i$. By Lemma 8, we know that $\vec{y} \cdot \vec{w} = \sum_X Pr[S' = X]\vec{y}(X) \cdot (\hat{w}_X, 0_{-X})$.

By Observation 9, we know that $\mathcal{A}'(\vec{w})_i = \sum_X Pr[S' = X|i \in S']A_X((\hat{w}_X, 0_{-X}))$, and therefore $\mathcal{A}'(\vec{w}) \cdot \vec{w} = \sum_X Pr[S' = X]A_X((\hat{w}_X, 0_{-X})) \cdot (\hat{w}_X, 0_{-X})$ as well by Lemma 8. By definition, A_X is an exact optimization algorithm for $Q(X)$, and $\vec{y}(X) \in Q(X)$, so we clearly have $A_X((\hat{w}_X, 0_{-X})) \cdot (\hat{w}_X, 0_{-X}) \geq \vec{y}(X) \cdot (\hat{w}_X, 0_{-X})$ for all X. Therefore, $\mathcal{A}'(\vec{w}) \cdot \vec{w} \geq \vec{y} \cdot \vec{w}$.

As this holds for any $\vec{y} \in P'$, the proposition is proved. □

6.3.2 Every point in P is close to some point in P'

In this section, we use the following Theorem due to Hoeffding [Hoe63]. We also w.l.o.g. assume that each $Q(S) \subseteq [0,1]^d$ (recall that we originally assumed that $Q(S) \subseteq [0, \text{poly}(d)]^d$. This further assumption simply costs us an extra poly(d) factor, but makes analysis simpler).

Theorem 12. *([Hoe63]) Let $X_1, \ldots, X_n$ be independent random variables in* $[0,1]$ *and let* $X = \sum_i X_i/n$. *Then* $Pr[|X - \mathbb{E}[X]| > t \leq 2e^{-2t^2 n}$.

We first show that a fixed $\vec{y} \in P$ is likely to be close to some $\vec{y} \in P'$, so long as enough samples are taken for D'. This will be a consequence of two simple lemmas: the first shows that, conditioned on enough S' in the support of D' containing i, y_i is likely to be close to y'_i. The second shows that we are likely to sample enough S' containing i. Following the work in Section 6.3.1, it's easy to see now that any $\vec{y} \in P$ can also be written using $\vec{y}(X) \in Q(X)$ and $y_i = \sum_X Pr[S = X|i \in S]y(X)_i$. We call such $\vec{y}(X)$ an *implementation* (with respect to D, if $\vec{y} \in P$, or with respect to D' if $\vec{y} \in P'$). What we will show is that the implementation of $\vec{y}$ with respect to D bears witness that some $\vec{y}\,' \in P$ is very close to $\vec{y}$. For the remainder of this section, for a given $\vec{y}$, define $\vec{y}\,'$ to have the same implementation as $\vec{y}$, but with respect to D' instead of D.

Lemma 9. *Fix $i \in [d]$ and any $\vec{y} \in P$. Condition on D' so that $|\{S' \in support(D')|i \in S'\}| = x$. Then over the randomness in generating D' (conditioned on this), we have* $Pr[|y'_i - y_i| > t] \leq 2e^{-2t^2 x}$.

Proof. Let S_j be the j^{th} subset in the support of D' with $i \in S_j$. And let $X_j = y(X_j)_i$. Then $y'_i = \sum_j X_j/x$, the X_j are independent and in $[0,1]$, and each $\mathbb{E}[X_j] = y_i$. Therefore, Hoeffding's inequality immediately yields the lemma. □

In the lemma below, recall that $b(P)$ is the sample complexity of P, which is $\max_i\{1/Pr[i \in S]\}$ for some corresponding sample-optimization algorithm $\mathcal{A}$.

Lemma 10. *Fix $i \in [d]$, and let $xb(P)$ samples be taken for D'. Let N_i denote the number of sets in the support of D' containing i. Then:*

$$Pr[N_i \leq x/2] \leq 2e^{-\frac{x}{2b(P)}}$$

Proof. Let X_j be the indicator random variable of whether or not the j^{th} sample for D' contains i. Then $N_i = \sum_j X_j$, and $\mathbb{E}[N_i] \geq x$. So in order to have $N_i \leq x/2$, we would also need $\frac{N_i}{xb(P)} \leq \mathbb{E}[\frac{N_i}{xb(P)}] - \frac{1}{2b(P)}$. Plugging this into the Hoeffding inequality yields the desired bound. □

Corollary 6. *Fix any $\vec{y} \in P$, and let $2xb(P)^2$ samples be taken for D'. Then:*

$$Pr[|\vec{y} - \vec{y}\,'|_\infty > t] \leq 4de^{-t^2 x}$$

Proof. Lemma 10 and a union bound guarantees that we will have all $N_i > x$ with probability at least $1 - 2de^{-x}$. A union bound combined with Lemma 9, conditioned on this, guarantees that for all i, $|y_i' - y_i| \leq t$ with probability at least $1 - 2de^{-2t^2 x}$. Taking a union bound over both of these events proves the corollary. □

Corollary 7. *Let $2xb(P)^2 d\ln(1/t)$ samples be taken for D'. Then with probability at least $1 - 4de^{-t^2 x}$, all $\vec{y} \in P$ have some $\vec{y}\,' \in P'$ satisfying $|\vec{y} - \vec{y}\,'|_\infty \leq 2t$.*

Proof. Consider an t-ℓ_∞ cover of P such that every point in P is within ℓ_∞ distance t of a point in the cover. There is certainly a cover that uses at most $\left(\frac{1}{t}\right)^d$ 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 $\vec{z} \in P'$, such that $|\vec{x} - \vec{z}|_\infty \leq 2t$, then clearly for every point $\vec{y}$ in P, there is a point $\vec{y}\,' \in P'$ such that $|\vec{y} - \vec{y}\,'|_\infty \leq 2t$ by the triangle inequality. So we simply take a union bound over all $e^{-d\ln t}$ points in the cover and apply Corollary 6 to conclude the proof. □

6.3.3 Every point in P' is close to some point in P

In the previous section, we showed that for all $\vec{y} \in P$ there is a nearby $\vec{y}\,' \in P'$ over the randomness in the choice of D'. In this section, we want to show the other direction, namely that for any $\vec{y}\,' \in P'$ there is a nearby $\vec{y} \in P$. However, it is not clear how to use the randomness over the choice of D' to prove this, as for any $\vec{y}\,' \in P'$, the implementation of $\vec{y}\,'$ is heavily dependent on D' itself, which is the object whose randomness we hope to use. To go around this circularity we show that after fixing the number of samples for D', but before actually selecting them, there are not too many implementations that could possibly yield a corner of P'. So we aim to take a union bound over all such implementations, showing that they yield vectors that are close with respect to D and D'. As every point in P' is a convex combination of the corners of P', this suffices to prove the desired claim. Our starting point is the following observation.

Lemma 11. *Let N be such that all corners of all $Q(S)$ have coordinates that are integer multiples of $1/N$, and x be the number of samples taken for D'. Then every corrner of P' has bit complexity $O(\log x + \log N)$.*

Proof. Recall that every corner $\vec{y}\,'$ of P' has a corresponding implementation such that $y'_i = \sum_X Pr[S' = X | i \in S'] y(X)_i$, where each $\vec{y}(X)$ is a corner of $Q(X)$. So each $y(X)_i$ is an integer multiple of $1/N$, and each probability is an integer multiple of $1/x$ because D' is a uniform distribution over x sets. So each y'_i will have a denominator of xN and a numerator at most xN, and therefore has bit complexity $O(\log x + \log N)$. □

Given Lemma 11 the corners of P' belong to a set of at most $2^{O(d(\log x + \log N))}$ vectors, independent of D'. Still, the implementation of those vectors may vary depending on D'. We show that this can be mitigated by appealing to the direction in which each corner is an extreme point.

Lemma 12. *Suppose that $\vec{y}$ is the corner of a closed convex region P' and that the corners of P' have bit complexity at most b. Then there exists a $\vec{w}$ of bit complexity* $\text{poly}(b, d)$ *such that* $\vec{y} = \text{argmax}_{\vec{x} \in P'}\{\vec{x} \cdot \vec{w}\}$.

Proof. Let P'_0 denote the convex hull of all corners of P' excluding $\vec{y}$, and consider solving (forgetting about computational efficiency, this is just an existence argument) the linear program of

Figure 6-2 from Section 6.1 using P'_0 as the closed convex region and $\vec{y}$ as point to test. Clearly, $\vec{y} \notin P'_0$, so we will necessarily find a violating hyperplane. That is, we will find a direction $\vec{w}$ such that $\vec{w} \cdot \vec{y} > \max_{\vec{x} \in P'_0}\{\vec{x} \cdot \vec{w}\}$. As exact optimization algorithms always output corners, every hyperplane ever considered by the linear program has coefficients of bit complexity b. Therefore, by Theorem 1, the output vector $\vec{w}$ has bit complexity poly(b, d). □

With Lemma 12, we know now that every corner $\vec{y}\,'$ of P' has an associated direction $\vec{w}$ of poly(b, d) bit complexity and $\vec{y}\,'$ can be implemented by having $\vec{y}\,'(X) = \text{argmax}_{\vec{x} \in Q}\{\vec{x} \cdot (\hat{w}_X, 0_{-X})\}$ for all X. As $Pr[i \in S']$ is an integer multiple of $1/x$ for all i, $\hat{w}$ also has bit complexity poly$(b, d, \log x)$. Therefore, before we have sampled D', but after we have chosen $|D'|$, there is a fixed set of at most $2^{\text{poly}(d,b,\log x)}$ implementations that can possibly result in a corner of D'. From here, we are ready to take a union bound over all such implementations and complete the proof.

Corollary 8. *Let $2xb(P)^2$ samples be taken for D' and $x = \text{poly}(d, t, b(P)^2, b)$ be sufficiently large. Then with probability at least $1 - 4de^{-t^2x/2}$, for all $\vec{y}\,' \in P'$, there exists a $\vec{y} \in P$ with $|\vec{y} - \vec{y}\,'|_\infty \leq t$. Furthermore, the same implementation that yields $\vec{y}\,'$ with respect to D' yields $\vec{y}$ with respect to D.*

Proof. We know from Corollary 6 that for any fixed implementation of $\vec{y} \in P$ with respect to D, that same implementation yields a $\vec{y}\,' \in P'$ with $|\vec{y} - \vec{y}\,'|_\infty \leq t$ with probability at least $1 - 4de^{-t^2x}$. We now want to take a union bound over all $2^{\text{poly}(d,b,\log(2xb(P)^2)}$ implementations that can possibly result in corners of P' and claim that they all satisfy this property. Notice that both terms get larger with x: as we decrease the probability of error, we need to take a union bound over more and more implementations. However, notice that the exponent of the error probability increases linearly with x, while the exponent of the number of terms we need to take a union bound over grows polylogarithmically in x. Therefore, if we let x grow sufficiently large (polynomial in $t, d, b, b(P)$ suffices), we will have poly$(d, b, \log(2xb(P)^2) < t^2x/2$. When this happens, we get the desired guarantee. □

Proof of Theorem 6: That P' is a sample-optimization region is an immediate consequence of Proposition 8 from Section 6.3.1. Taking $t = \epsilon$, the first listed property of P' is an immediate consequence of Corollary 7 from Section 6.3.2. The second property is an immediate consequence of Corollary 8 above (again taking $t = \epsilon$). ■

Chapter 7

(α, β)-Approximation Algorithms for Makespan and Fairness

Here, we design algorithms for scheduling unrelated machines with costs and fair allocation of indivisible goods with costs and show that they imply mechanisms for scheduling unrelated machines and fair allocation of indivisible goods using the reduction of Chapter 5.

Theorems 10 and 11 in Chapter 5 are immediate corollaries of Theorem 7 combined with Theorem 13 or Theorem 15.

7.1 Preliminaries

Here we define the problems of scheduling unrelated machines with costs and fair allocation of indivisible goods with costs (noting that their versions without costs may simply set all costs to 0 in the definition).

Scheduling Unrelated Machines with Costs. There are k machines and m indivisible jobs. Each machine i can process job j in time $t_{ij} \geq 0$. Additionally, processing job j on machine i costs c_{ij} units of currency, where c_{ij} is unrestricted and in particular could be negative. An assignment of jobs to machines is a km-dimensional vector $\vec{x}$ such that $x_{ij} \in \{0, 1\}$, for all i and j, where job j is assigned to machine i iff $x_{ij} = 1$. We denote by $M(\vec{x}) = \max_i\{\sum_j x_{ij}t_{ij}\}$ the makespan of an assignment, by $F(\vec{x}) = \min_i\{\sum_j x_{ij}t_{ij}\}$ the fairness of an assignment, and by $C(\vec{x}) = \sum_i \sum_j x_{ij}c_{ij}$ the cost of an assignment. In the makespan minimization problem, an assignment is valid iff $\sum_i x_{ij} = 1$ for all jobs j, while in the fairness maximization problem, an assignment is valid iff $\sum_i x_{ij} \leq 1$. To avoid carrying these constraints around, we use the convention that $M(\vec{x}) = \infty$, if $\sum_i x_{ij} \neq 1$ for some j, and $F(\vec{x}) = -\infty$, if $\sum_i x_{ij} > 1$ for some j. It will also be useful for analysis purposes to consider fractional assignments of jobs, which relax the constraints $x_{ij} \in \{0, 1\}$ to $x_{ij} \in [0, 1]$. This corresponds to assigning an x_{ij}-fraction of job j to machine i for all pairs (i, j). Notice that $M(\vec{x})$, $F(\vec{x})$ and $C(\vec{x})$ are still well-defined for fractional assignments.

The goal of makespan minimization with costs is to find an assignment $\vec{x} \in \{0, 1\}^{km}$ satisfying $M(\vec{x}) + C(\vec{x}) = \min_{\vec{x}\,' \in \{0,1\}^{km}}\{M(\vec{x}\,') + C(\vec{x}\,')\}$. In the language of GOOP, this is GOOP($\{0, 1\}^{km}$, {additive functions with non-negative coefficients}, M). The processing times $\vec{t_i}$ correspond to the functions g_i that are input to GOOP, and the costs $\vec{c}$ corresponds to the function f. For $\alpha \geq 1 \geq \beta$, an (α,β)-approximation for this problem is an assignment $\vec{x} \in \{0, 1\}^{km}$ with $\beta M(\vec{x}) + C(\vec{x}) \leq \alpha \min_{\vec{x}\,' \in \{0,1\}^{km}}\{M(\vec{x}\,') + C(\vec{x}\,')\}$.

The goal of fairness maximization with costs is to find an assignment $\vec{x} \in \{0, 1\}^{km}$ satisfying $F(\vec{x}) + C(\vec{x}) = \max_{\vec{x}\,' \in \{0,1\}^{km}}\{F(\vec{x}\,') + C(\vec{x}\,')\}$. In the language of GOOP, this is GOOP($\{0, 1\}^{km}$, {additive functions with non-negative coefficients}, F). Again, the processing times $\vec{t_i}$ correspond to the functions g_i that are input to GOOP, and the costs $\vec{c}$ corresponds to the function f. For $\alpha \leq 1 \leq \beta$, an (α,β)-approximation for this problem is an assignment $\vec{x} \in \{0, 1\}^{km}$ with $\beta F(\vec{x}) + C(\vec{x}) \geq \alpha \max_{\vec{x}\,' \in \{0,1\}^{km}}\{F(\vec{x}\,') + C(\vec{x}\,')\}$.

Note that in the case of maximizing fairness, sometimes the jobs are thought of as gifts and the machines are thought of as children (and the problem is called the Santa Claus problem). In this case, it makes sense to think of the children as having value for the gifts (and preferring more value to less value) instead of the machines having processing time for jobs (and preferring less processing time to more). For ease of exposition, we will stick to the jobs/machines interpretation, although our results extend to the gifts/children interpretation as well.

We conclude this section by noting that both makespan minimization with costs and fairness maximization with costs are NP-hard to approximate within any finite factor. For makespan with costs, this can be seen via a simple modification of an inapproximability result given in [LST87]. For the problem of scheduling unrelated machines, they construct instances with integer-valued makespan that is always ≥ 3 and such that it is NP-hard to decide whether the makespan is 3 or ≥ 4. We can modify their instances to scheduling unrelated machines with costs instances by giving each job a cost of $\frac{z-3}{2n+m}$ on every machine for an arbitrary $z > 0$. Then the total cost of any feasible solution is exactly $z - 3$. So their proof immediately shows that it is NP-hard to determine if these instances have optimal makespan + cost that is z or $\geq 1 + z$. Since z was arbitrary, this shows that no finite approximation factor is possible.

For fairness with costs, Bezakova and Dani [BD05] present a family of max-min fairness instances such that it is NP-hard to distinguish between OPT ≥ 2 and OPT ≤ 1. To each of these instances add a special machine and a special job such that the processing time and cost of the special machine for the special job are 2 and -1 respectively, while the processing time and cost of the special machine for any non-special job or of any non-special machine for the special job are 0 and 0 respectively. Also, assign 0 cost to any non-special machine non-special job pair. In the resulting max-min fairness with costs instances it is NP-hard to distinguish between OPT ≥ 1 and OPT $= 0$, hence no finite approximation is possible.

7.2 Algorithmic Results

In this section we develop bicriterion algorithmic results for minimizing makespan and maximizing fairness. Additionally, we state a general theorem that is useful in developing (α,β)-approximation algorithms via algorithms that round fractional solutions. In particular, this theorem allows us to get a $(\frac{1}{2}, \tilde{O}(\sqrt{k}))$-approximation for fairness based on the algorithm of Asadpour and Saberi [AS07]. We begin by stating our (α,β)-guarantees.

Theorem 13. *There is a polynomial-time $(1, \frac{1}{2})$-approximation for minimizing makespan with costs on unrelated machines. The algorithm is based on a rounding theorem of Shmoys and Tardos [ST93b].*

Proof. Shmoys and Tardos show that if the linear program of Figure 7-1 outputs a feasible fractional solution, then it can be rounded to a feasible integral solution without much loss. We will refer to this linear program as $LP(t)$ for various values of t.

Variables:

- x_{ij}, for all machines i and jobs j denoting the fractional assignment of job j to machine i.
- T, denoting the maximum of the makespan and the processing time of the largest single job used.

Constraints:

- $\sum_{i=1}^{k} x_{ij} = 1$, for all j, guaranteeing that every job is assigned.
- $\sum_{j=1}^{m} t_{ij}x_{ij} \leq T$, for all i, guaranteeing that the makespan is at most T.
- $x_{ij} \geq 0$, for all i, j.
- $x_{ij} = 0$ for all i, j such that $t_{ij} > t$, guaranteeing that no single job has processing time larger than t.
- $T \geq t$.

Minimizing:

- $\sum_{i,j} c_{ij}x_{ij} + T$, (almost) the makespan plus cost of the fractional solution.

Figure 7-1: $LP(t)$.

Theorem 14. *([ST93b]) Any feasible solution to $LP(t)$ can be rounded to a feasible integral solution in polynomial time with makespan at most $T + t$ and cost at most $C = \sum_{i,j} c_{ij}x_{ij}$.*

With Theorem 14 in hand, we can now design a $(1, 1/2)$-approximation algorithm. Define $\hat{M}(\cdot)$ as the modified makespan of an assignment to be $\hat{M}(\vec{x}) = \max_{i,j|x_{ij}>0}\{M(x), t_{ij}\}$. In other words, $\hat{M}(\vec{x})$ is the larger of the makespan and the processing time of the largest single job that is fractionally assigned. Note that for any $\vec{x} \in \{0,1\}^{km}$ that $M(\vec{x}) = \hat{M}(\vec{x})$. Now consider solving $LP(t)$ for all km possible values of t, and let $\vec{x}^*$ denote the best solution among all feasible solutions output. The following lemma states that $\vec{x}^*$ performs better than the integral optimum.

Lemma 13. *Let $\vec{x}^*$ denote the best feasible solution output among all km instances of $LP(t)$, and let $\vec{y}$ denote the integral solution minimizing makespan plus cost. Then $\hat{M}(\vec{x}^*) + C(\vec{x}^*) \leq M(\vec{y}) + C(\vec{y})$.*

Proof. Some job assigned in $\vec{y}$ has the largest processing time, say it is t. Then $\vec{y}$ is a feasible solution to $LP(t)$, and will have value $\hat{M}(\vec{y}) + C(\vec{y})$. $\vec{x}^*$ therefore satisfies $\hat{M}(\vec{x}^*) + C(\vec{x}^*) \leq \hat{M}(\vec{y}) + C(\vec{y})$. As $\vec{y}$ is an integral solution, we have $\hat{M}(\vec{y}) = M(\vec{y})$, proving the lemma. □

Comining Lemma 13 with Theorem 14 proves Theorem 13. Consider the algorithm that solves $LP(t)$ for all km values of t and outputs the fractional solution $\vec{x}^*$ that is optimal among all feasible solutions found. By Lemma 13, $\vec{x}^*$ is at least as good as the optimal integral solution. By Theorem 14, we can continue by rounding $\vec{x}^*$ in polynomial time to an integral solution $\vec{x}$ satisfying $\frac{1}{2}M(\vec{x}) + C(\vec{x}) \leq \hat{M}(\vec{x}^*) + C(\vec{x}^*) \leq M(\vec{y}) + C(\vec{y})$. □

We now move on to fairness. We postpone the proof of Theorem 15 to the end of the section.

Theorem 15. *There is a polynomial-time algorithm for maximizing fairness with costs on unrelated machines with the following guarantees:*

- *A $(\frac{1}{2}, \tilde{O}(\sqrt{k}))$-approximation based on an algorithm of Asadpour and Saberi [AS07].*
- *A $(1, m - k + 1)$-approximation based on an algorithm of Bezakova and Dani [BD05].*

Next, we state a theorem useful in the analysis of algorithms that round fractional solutions. In the theorem statement below, $\vec{x} \in [0,1]^{km}$ denotes a fractional assignment of jobs to machines, and $\vec{y}$ denotes a randomly sampled assignment in $\{0,1\}^{km}$ according to some rounding procedure. Note that when we write $F(\vec{y})$, we mean the expected value of the random variable $F(\vec{y})$, and *not* the fairness computed with respect to the fractional assignment $\mathbb{E}[y_{ij}]$. Finally, $\vec{v} \in \{0,1\}^{km}$ denotes the integral allocation that maximizes $C(\vec{w})$ over all $\vec{w} \in \{0,1\}^{km}$. In other words, $\vec{v}$ assigns each job to the machine with the highest non-negative cost, if one exists (and nowhere if none exists). We emphasize again that Theorem 16 is what enables the first bullet in Theorem 15 above, and will likely have applications to the analysis of other potential (α,β)-approximation algorithms. We also note that Theorem 16 applies to any maximization objective, not just fairness.

Theorem 16. *Let $\vec{x} \in [0,1]^{km}$ be a fractional assignment of jobs to machines that is an (α,β)-approximation with respect to the optimal integral assignment (that is, $\beta F(\vec{x}) + C(\vec{x}) \geq \alpha OPT$), and $\vec{y}$ a random variable of assignments supported on $\{0,1\}^{km}$ satisfying $z \cdot F(\vec{y}) \geq F(\vec{x})$, for some $z \geq 1$, and $\mathbb{E}[y_{ij}] \leq x_{ij}$ for all i, j. Then for any $\gamma \in [0,1]$, at least one of the following is true:*

- $z \cdot \beta F(\vec{y}) + C(\vec{y}) \geq \gamma(\beta F(\vec{x}) + C(\vec{x})) \geq \gamma \cdot \alpha OPT$. *That is, $\vec{y}$ is a $(\gamma\alpha, z\beta)$-approximation.*

- *Or* $F(\vec{v}) + C(\vec{v}) \geq (1-\gamma)(\beta F(\vec{x}) + C(\vec{x})) \geq (1-\gamma) \cdot \alpha OPT$. *That is, $\vec{v}$ is a $((1-\gamma)\alpha, 1)$-approximation. $\vec{v}$ is the assignment maximizing $C(\vec{w})$ over all feasible $\vec{w} \in \{0,1\}^{km}$.*

Proof. We can break the cost of $\vec{x}$ into $C(\vec{x}) = C^+(\vec{x}) + C^-(\vec{x})$, where $C^+(\vec{x})$ denotes the portion of the cost due to jobs assigned to machines with positive cost, and $C^-(\vec{x})$ denotes the portion of the cost due to jobs assigned to machines with negative cost. As $\vec{v}$ assigns all jobs to the machine with largest positive cost, we clearly have $C^+(\vec{v}) \geq C^+(\vec{x})$ and $C^-(\vec{v}) = 0$ (but may have $F(\vec{v}) = 0$). Furthermore, as $\mathbb{E}[y_{ij}] \leq x_{ij}$ for all i, j, we clearly have $C^-(\vec{y}) \geq C^-(\vec{x})$ (but may have $C^+(\vec{y}) = 0$).

So there are two cases to consider. Maybe $\beta F(\vec{x}) + C^-(\vec{x}) \geq \gamma(\beta F(\vec{x}) + C(\vec{x}))$. In this case, we clearly have $z \cdot \beta F(\vec{y}) + C(\vec{y}) \geq \beta F(\vec{x}) + C^-(\vec{x}) \geq \gamma(\beta F(\vec{x}) + C(\vec{x}))$, and the first possibility holds. The other case is that maybe $C^+(\vec{x}) \geq (1-\gamma)(\beta F(\vec{x}) + C(\vec{x}))$, in which case we clearly have $F(\vec{v}) + C(\vec{v}) \geq C^+(\vec{x}) \geq (1-\gamma)(\beta F(\vec{x}) + C(\vec{x}))$, and the second possibility holds. □

With Theorem 16, we may now prove Theorem 15. We begin by describing our algorithm modifying that of Asadpour and Siberi, which starts by solving a linear program known as the configuration LP. We modify the LP slightly to minimize fairness plus cost, but this does not affect the ability to solve this LP in polynomial time via the same approach used by Bansal and Sviridenko [BS06].[1] The modified configuration LP is in Figure 7-2. Note that T is a parameter, and for any T we call the instantiation of the configuration LP $CLP(T)$. A configuration is a set S of jobs. A configuration S is said to be "valid" for machine i if $\sum_{j \in S} t_{ij} \geq T$, or if S contains a single job with $t_{ij} \geq T/\sqrt{k}\log^3(k)$. Call the former types of configurations "small" and the latter "big." $S(i, T)$ denotes the set of all configurations that are valid for machine i.

Step one of the algorithm solves $CLP(T)$ for all $T = 2^x$ for which the fairness of the optimal solution could possibly be between 2^x and 2^{x+1}. It's clear that there are only polynomially many (in the bit complexity of the processing times and k and m) such x. Let $\vec{x}(T)$ denote the solution found by solving $CLP(T)$ (if one was found at all). Then define $\vec{x}^* = \text{argmax}_T\{2T + C(\vec{x}(T))\}$. We first claim that $\vec{x}^*$ is a good fractional solution.

[1]Note that this is non-trivial, as the LP has exponentially-many variables. The approach of Bansal and Sviridenko is to solve the dual LP via a separation oracle which requires solving a knapsack problem.

Variables:

- $x_{i,S}$, for all machines i and configurations S denoting the fractional assignment of configuration S to machine i.

Constraints:

- $\sum_{S \in S(i,T)} x_{i,S} = 1$, for all i, guaranteeing that every machine is fractionally assigned a valid configuration with weight 1.
- $\sum_i \sum_{S|j \in S} x_{i,S} \leq 1$, for all j, guaranteeing that no job is fractionally assigned with weight more than 1.
- $x_{i,S} \geq 0$, for all i, S.

Maximizing:

- $\sum_i \sum_{S \in S(i,T)} x_{i,S} \sum_{j \in S} c_{ij}$, the cost of the fractional solution $\vec{x}$.

Figure 7-2: (a modification of) The configuration LP parameterized by T.

Lemma 14. *Let OPT be the fairness plus cost of the optimal integral allocation. Then $2F(\vec{x}^*) + C(\vec{x}^*) \geq OPT$.*

Proof. Whatever the optimal integral allocation, $\vec{z}$ is, it has some fairness $F(\vec{z})$. For $T = 2^x$ satisfying $F(\vec{z}) \in [T, 2T)$, $\vec{z}$ is clearly a feasible solution to $CLP(T)$, and therefore we must have $C(\vec{x}(T)) \geq C(\vec{z})$. As we also clearly have $2T \geq F(\vec{z})$ by choice of T, we necessariliy have $2F(\vec{x}(T)) + C(\vec{x}(T)) \geq OPT$. As $\vec{x}^*$ maximizes $2F(\vec{x}(T)) + C(\vec{x}(T))$ over all T, it satisfies the same inequality as well. □

From here, we will make use of Theorem 16: either we will choose the allocation $\vec{v}$ that assigns every job to the machine with the highest non-negative cost, or we'll round $\vec{x}^*$ to $\vec{y}$ via the procedure used in [AS07]. We first state the rounding algorithm of [AS07].

1. Make a bipartite graph with k nodes (one for each machine) on the left and m nodes (one for each job) on the right.
2. For each machine i and job j, compute $x_{ij} = \sum_{S \ni j} x_{i,S}$. If $t_{ij} \geq T/\sqrt{k}\log^3 k$, put an edge of weight x_{ij} between machine i and job j. Call the resulting graph $\mathcal{M}$.

3. For each node v, denote by m_v the sum of weights of edges incident to v.

4. Update the weights in $\mathcal{M}$ to remove all cycles. This can be done without decreasing $C(\vec{x})$ or changing m_v for any v, and is proved in Lemma 15.

5. Pick a random matching M in $\mathcal{M}$ according to Algorithm 2 of [AS07]. Each edge (i, j) will be included in M with probability exactly x_{ij}, and each machine i will be matched with probability exactly m_i.

6. For all machines that were unmatched in M, select small configuration S with probability $x_{i,S}/m_i$.

7. For all jobs that were selected both in the matching stage and the latter stage, award them just to whatever machine received them in the matching. For all jobs that were selected only in the latter stage, choose a machine uniformly at random among those who selected it. Throw away all unselected jobs.

Before continuing, let's prove that we can efficiently remove cycles without decreasing the cost or changing any m_v.

Lemma 15. *Let $\mathcal{M}$ be a bipartite graph with an edge of weight x_{ij} between node i and node j, and denote by m_v the sum of weights of edges incident to v. Let also each edge have a cost, c_{ij}. Then we can modify the weights of $\mathcal{M}$ in poly-time so that $\mathcal{M}$ is acyclic, without decreasing $\sum_{i,j} x_{ij}c_{ij}$ or changing any m_v.*

Proof. Consider any cycle $e_1, \ldots, e_{2x}$. For $e = (i, j)$, denote by $c(e) = c_{ij}$ and $x(e) = x_{ij}$. Call the odd edges those with odd subscripts and the even edges those with even subscripts. W.l.o.g. assume that the odd edges have higher total cost. That is, $\sum_{z=1}^{x} c(e_{2z-1}) \geq \sum_{z=1}^{x} c(e_{2z})$. Let also $\epsilon = \min_z x(e_{2z})$. Now consider decreasing the weight of all even edges by ϵ and increasing the weight of all odd edges by ϵ. Clearly, we have not decreased the cost. It is also clear that we have not changed m_v for any v. And finally, it is also clear that we've removed a cycle (by removing an edge). So we can repeat this procedure a polynomial number of times and result in an acyclic graph. □

Now, let $\vec{x}$ denote the fractional assignment obtained after removing cycles in $\mathcal{M}$. Then it's clear that $2F(\vec{x}) + C(\vec{x}) \geq OPT$. If we let $\vec{y}$ denote the randomized allocation output at the end of the procedure, it's also clear that $\mathbb{E}[y_{ij}] \leq x_{ij}$ for all i, j. This is because if there were never any conflicts (jobs being awarded multiple times), we would have exactly $\mathbb{E}[y_{ij}] = x_{ij}$. But because of potential conflicts, $\mathbb{E}[y_{ij}]$ can only decrease. Asadpour and Siberi show the following theorem about the quality of $\vec{y}$:

Theorem 17. *([AS07]) With probability $1 - o(1)$, the fairness of the allocation output by the procedure is at least $F(\vec{x})/320\sqrt{k}\log^3 k$. This implies that $F(\vec{y}) \in \tilde{\Omega}(F(\vec{x})/\sqrt{k})$.*

And now we are ready to make use of Theorem 16.

Proposition 9. *It is either the case that assigning every job to the machine with highest cost is a $(\frac{1}{2}, 1)$-approximation (which is a traditional $\frac{1}{2}$-approximation), or the $\vec{y}$ output by the algorithm above is a $(\frac{1}{2}, \tilde{O}(\sqrt{k}))$-approximation.*

Proof. After removing cycles, we have a fractional solution $\vec{x}$ that is a $(1, 2)$-approximation. By using the randomized procedure of Asadpour and Siberi, we get a randomized $\vec{y}$ satisfying $\mathbb{E}[y_{ij}] \leq x_{ij}$ for all i, j and $\tilde{O}(\sqrt{k})F(\vec{y}) \geq F(\vec{x})$. Therefore, taking $\gamma = 1/2$, Theorem 16 tells us that either assigning every job to the machine with highest non-negative cost yields a $\frac{1}{2}$-approximation, or $\vec{y}$ is a $(\frac{1}{2}, \tilde{O}(\sqrt{k}))$-approximation. □

We conclude this section by proving that the $(m - k + 1)$-approximation algorithm of Bezakova and Dani for fairness can be modified to be a $(1, m-k+1)$-approximation for fairness plus costs. The algorithm is fairly simple: for a fixed T, make the following bipartite graph. Put k nodes on the left, one for each machine, and m nodes on the right, one for each job. Put an additional $m - k$ nodes on the left for dummy machines. Put an edge from every job node j to every dummy machine node of weight $\max_i\{0, c_{ij}\}$, and an edge from every job node to every real machine node i of weight c_{ij} *only if* $t_{ij} \geq T$. Then find the maximum weight matching in this graph. For every job that is matched to a real machine, assign it there. For every job that is assigned to a dummy machine, assign it to the machine with the maximum non-negative cost (or nowhere if they're all negative). Call this assignment A_T. Denote $A^* = \text{argmax}_T\{(m - k + 1)T + C(A_T)\}$. Finally, let V denote the allocation that just assigns every job to the machine of highest cost. If $F(V)+C(V) \geq (m-k+1)F(A^*)+C(A^*)$, output V. Otherwise, output A^*.

Proposition 10. *The algorithm above finds an allocation A satisfying $(m-k+1)F(A)+C(A) \geq OPT$. That is, A is a $(1, m-k+1)$-approximation.*

Proof. Consider the optimal assignment X. We either have $F(X) = 0$, or $F(X) > 0$. If $F(X) = 0$, then clearly $X = V$. If $F(X) > 0$, then every machine is awarded at least one job, but at most $m-k+1$. For each machine i, define $j(i)$ to be the job assigned to i with the highest processing time. Except for $j(i)$, reassign all other jobs to the machine with the highest non-negative cost. This can only increase the cost, and will not hurt the fairness by more than a factor of $m-k+1$. So this solution, X', clearly has $(m-k+1)F(X')+C(X') \geq OPT$. Futhermore, X' corresponds to a feasible matching when $T = F(X')$. Whatever solution A_T is found instead clearly has fairness at least T and cost at least $C(X')$. So A_T, and therefore also A^*, is a $(1, m-k+1)$-approximation.

So in conclusion, either $F(X) > 0$, in which case A^* is a $(1, m-k+1)$-approximation, or $F(X) = 0$, in which case $F(V)+C(V) = OPT$. So if we ever output V, we actually have $V = OPT$. If we output A^*, then either $F(X) > 0$, or $(m-k+1)F(A^*)+C(A^*) \geq F(V)+C(V) = OPT$. In both cases, A^* is a $(1, m-k+1)$-approximation. □

Proof of Theorem 15: Part 1) is proved in Proposition 9, and part 2) is proved in Proposition 10. ■

Chapter 8

Revenue: Hardness of Approximation

In this chapter we provide our framework for proving hardness of approximation and use it to show that revenue maximization is NP-hard to approximate within any polynomial factor for a single monotone submodular agent. We describe briefly first the prior work on the computational hardness of revenue maximization.

Briest shows that for a single unit-demand agent, under well-believed complexity theory assumptions, it is computationally hard to find any deterministic auction whose revenue is within a poly(n^ϵ) factor of optimal for some $\epsilon > 0$ [Bri08]. Papadimitriou and Pierrakos show that for three agents whose values for a single item are correlated that finding the optimal deterministic auction is APX-hard. Daskalakis, Deckelbaum and Tzamos show that, unless $P^{\#P} \subseteq ZPP$, one cannot find the optimal auction for a single additive agent whose value for each of m items is sampled independently from a distribution of support 2 in time poly(m) [DDT14]. Similarly, Chen et. al. show that it is NP-hard to find the optimal deterministic mechanism for a single unit-demand agent whose value for each of m items is sampled independently from a distribution of support 3 [CDP$^+$14]. Lastly, Dobzinski, Fu, and Kleinberg show that it is NP-hard to find the optimal mechanism for a single agent whose value for subsets of m items is OXS [DFK11].[1]

In summary, prior work establishing hardness of approximation only does so for finding *deterministic* mechanisms [Bri08, PP11]. In both settings, the optimal randomized mechanism can be

[1]OXS valuations are a subclass of submodular functions.

found in polynomial time. The work of [DDT14, CDP+14] establishes computational hardness for finding an exact solution, and also aims for a stronger runtime that is polynomial in the number of items rather than polynomial in the number of potential types. Our results are therefore somewhat incomparable to these. Our techniques are most similar to those used in [DFK11], however their techniques are limited to showing only exact hardness, whereas we are able to show hardness of approximation.

In developing our framework, we make use of the following generalization of GOOP that we call the **S**olve **A**ny **D**ifferences **P**roblem (SADP). Essentially, SADP gives the solver the freedom to solve any one of several instances of GOOP. We repeat the definition of BMeD for revenue below, followed by the definition of SADP.

BMeD($\mathcal{F}, \mathcal{V}$): Input: A finite set of types T, and for each agent $i \in [m]$, a distribution $\mathcal{D}_i$ over T. Goal: Find a feasible (outputs an outcome in $\mathcal{F}$ with probability 1) BIC, IR mechanism M, that maximizes expected revenue, when k agents with types sampled from $\mathcal{D} = \times_i \mathcal{D}_i$ play M truthfully (with respect to all feasible, BIC, IR mechanisms). M is said to be an α-approximation to BMeD if its expected revenue is at least a α-fraction of the optimal obtainable expected revenue.

For our reduction, we will only require BMeD instances with $k = 1$.

SADP($\mathcal{F}, \mathcal{V}$): Given as input functions $f_j \in \mathcal{V}^\times$ ($1 \le j \le n$), find a feasible outcome $X \in \mathcal{F}$ such that there exists an index $j^* \in [n-1]$ such that:

$$f_{j^*}(X) - f_{j^*+1}(X) = \max_{X' \in \mathcal{F}} \{f_{j^*}(X') - f_{j^*+1}(X')\}.$$

X is said to be an α-approximation to SADP if there exists an index $j^* \in [n-1]$ such that:

$$f_{j^*}(X) - f_{j^*+1}(X) \ge \alpha \max_{X' \in \mathcal{F}} \{f_{j^*}(X') - f_{j^*+1}(X')\}.$$

8.1 Cyclic Monotonicity and Compatibility

We start with two definitions that discuss matching types to allocations. By this, we mean to form a bipartite graph with types on the left and (possibly randomized) allocations on the right. The weight of an edge between a type t and (possibly randomized) allocation X is exactly $t(X)$. So the weight of a matching in this graph is just the total welfare obtained by awarding each allocation to its matched type. So when we discuss the welfare-maximizing matching of types to allocations, we mean the max-weight matching in this bipartite graph. Below, cyclic monotonicity is a well-known definition in mechanism design with properties connected to truthfulness. Compatibility is a new property that is slightly stronger than cyclic monotonicity.

Definition 4. *(Cyclic Monotonicity) A list of (possibly randomized) allocations $(X_1, \dots, X_n)$ is said to be cyclic monotone with respect to $(t_1, \dots, t_n)$ if the welfare-maximizing matching of types to allocations is to match allocation X_i to type t_i for all i.*

Definition 5. *(Compatibility) We say that a list of types $(t_1, \dots, t_n)$ and a list of (possibly randomized) allocations $(X_1, \dots, X_n)$ are* compatible *if $(X_1, \dots, X_n)$ is cyclic monotone with respect to $(t_1, \dots, t_n)$, and for any $i < j$, the welfare-maximizing matching of types $t_{i+1}, \dots, t_j$ to (possibly randomized) allocations $X_i, \dots, X_{j-1}$ is to match allocation X_ℓ to type $t_{\ell+1}$ for all ℓ.*

Compatibility is a slightly stronger condition than cyclic monotonicity. In addition to the stated definition, it is easy to see that cyclic monotonicity also guarantees that for all $i < j$, the welfare-maximizing matching of types $t_{i+1}, \dots, t_j$ to allocations $X_{i+1}, \dots, X_j$ is to match allocation X_ℓ to type t_ℓ for all ℓ. Compatibility requires that the "same" property still holds if we shift the allocations down by one.

Now, we want to understand how the type space and allocations relate to expected revenue. Below, $\vec{t}$ denotes an ordered list of n types, $\vec{q}$ denotes an ordered list of n probabilities and $\vec{X}$ denotes an ordered list of n (possibly randomized) allocations. We denote by $Rev(\vec{t}, \vec{q}, \vec{X})$ the maximum obtainable expected revenue in a BIC, IR mechanism that awards allocation X_j to type t_j, and the agent's type is t_j with probability q_j. Proposition 11 provides an upper bound on Rev that holds in every instance. Proposition 12 provides sufficient conditions for this bound to be tight.

Proposition 11. *For all $\vec{t}, \vec{q}, \vec{X}$, we have:*

$$Rev(\vec{t}, \vec{q}, \vec{X}) \leq \sum_{\ell=1}^{n} \sum_{j \geq \ell} q_j t_\ell(X_\ell) - \sum_{\ell=1}^{n-1} \sum_{j \geq \ell+1} q_j t_{\ell+1}(X_\ell) \tag{8.1}$$

Proof. Let p_j denote the price charged in some BIC, IR mechanism that awards allocation X_j to type t_j. Then IR guarantees that:

$$p_1 \leq t_1(X_1)$$

Furthermore, BIC guarantees that, for all $j \leq n$:

$$\begin{aligned} t_j(X_j) - p_j &\geq t_j(X_{j-1}) - p_{j-1} \\ \Rightarrow p_j &\leq t_j(X_j) - t_j(X_{j-1}) + p_{j-1} \end{aligned}$$

Chaining these inequalities together, we see that, for all $j \leq n$:

$$p_j \leq t_1(X_1) + \sum_{\ell=2}^{j} \left(t_\ell(X_\ell) - t_\ell(X_{\ell-1})\right)$$

As the expected revenue is exactly $\sum_j p_j q_j$, we can rearrange the above inequalities to yield:

$$\begin{aligned} \sum_j p_j q_j &\leq t_1(X_1) \sum_j q_j + \sum_{\ell=2}^{n} \sum_{j \geq \ell} q_j \left(t_\ell(X_\ell) - t_\ell(X_{\ell-1})\right) \\ &\leq t_1(X_1) + \sum_{\ell=2}^{n} \sum_{j \geq \ell} q_j t_\ell(X_\ell) - \sum_{\ell=2}^{n} \sum_{j \geq \ell} q_j t_\ell(X_{\ell-1}) \\ &\leq \sum_{\ell=1}^{n} \sum_{j \geq \ell} q_j t_\ell(X_\ell) - \sum_{\ell=1}^{n-1} \sum_{j \geq \ell+1} q_j t_{\ell+1}(X_\ell) \end{aligned}$$

□

Proposition 12. *If $\vec{t}$ and $\vec{X}$ are compatible, then for all $\vec{q}$, Equation* (8.1) *is tight. That is,*

$$Rev(\vec{t}, \vec{w}, \vec{X}) = \sum_{\ell=1}^{n} \sum_{j \geq \ell} q_j t_\ell(X_\ell) - \sum_{\ell=1}^{n-1} \sum_{j \geq \ell+1} q_j t_{\ell+1}(X_\ell)$$

Proof. Proposition 11 guarantees that the maximum obtainable expected payment does not exceed the right-hand bound, so we just need to give payments that yield a BIC, IR mechanism whose expected payment is as desired. So consider the same payments used in the proof of Proposition 11:

$$\begin{aligned} p_1 &= t_1(X_1) \\ p_j &= t_j(X_j) - t_j(X_{j-1}) + p_{j-1},\ j \geq 2 \\ &= t_1(X_1) + \sum_{\ell=2}^{j} (t_\ell(X_\ell) - t_\ell(X_{\ell-1})) \end{aligned}$$

The exact same calculations as in the proof of Proposition 11 shows that the expected revenue of a mechanism with these prices is exactly the right-hand bound. So we just need to show that these prices yield a BIC, IR mechanism. For simplicity in notation, let $p_0 = 0$, $t_0 = 0$ (the 0 function), and X_0 be the null allocation (for which we have $t_j(X_0) = 0\ \forall j$). To show that these prices yield a BIC, IR mechanism, we need to show that for all $j \neq \ell$:

$$t_j(X_j) - p_j \geq t_j(X_\ell) - p_\ell$$

$$\Leftrightarrow t_j(X_j) - t_j(X_\ell) + p_\ell - p_j \geq 0$$

First, consider the case that $j > \ell$. Then:

$$\begin{aligned} t_j(X_j) - t_j(X_\ell) + p_\ell - p_j &= t_j(X_j) - t_j(X_\ell) - \sum_{z=\ell+1}^{j} (t_z(X_z) - t_z(X_{z-1})) \\ &= t_j(X_j) + \sum_{z=\ell+1}^{j} t_z(X_{z-1}) - t_j(X_\ell) - \sum_{z=\ell+1}^{j} t_z(X_z) \\ &= \sum_{z=\ell+1}^{j} t_z(X_{z-1}) - t_j(X_\ell) - \sum_{z=\ell+1}^{j-1} t_z(X_z) \end{aligned}$$

Notice now that $\sum_{z=\ell+1}^{j} t_z(X_{z-1})$ is exactly the welfare of the matching that gives X_{z-1} to t_z for all $z \in \{\ell+1, \ldots, j\}$. Similarly, $t_j(X_\ell) + \sum_{z=\ell+1}^{j-1} t_z(X_{z-1})$ is exactly the welfare of the matching that gives X_z to t_z for all $z \in \{\ell+1,\dots,j-1\}$ and gives X_ℓ to t_j. In other words, both sums represent the welfare of a matching between allocations in $\{X_\ell,\dots,X_{j-1}\}$ and types in $\{t_{\ell+1},\dots,t_j\}$. Furthermore, compatibility guarantees that the first sum is larger. Therefore, this term is positive, and t_j cannot gain by misreporting any t_ℓ for $\ell < j$ (including $\ell = 0$).

The case of $j < \ell$ is nearly identical, but included below for completeness:

$$\begin{aligned}
t_j(X_j) - t_j(X_\ell) + p_\ell - p_j &= t_j(X_j) - t_j(X_\ell) + \sum_{z=j+1}^{\ell} (t_z(X_z) - t_z(X_{z-1})) \\
&= t_j(X_j) + \sum_{z=j+1}^{\ell} t_z(X_z) - t_j(X_\ell) - \sum_{z=j+1}^{\ell} t_z(X_{z-1}) \\
&= \sum_{z=j}^{\ell} t_z(X_z) - t_j(X_\ell) - \sum_{z=j+1}^{\ell} t_z(X_{z-1})
\end{aligned}$$

Again, notice that $\sum_{z=j}^{\ell} t_z(X_z)$ is exactly the welfare of the matching that gives X_z to t_z for all $z \in \{j,\dots,\ell\}$. Similarly, $t_j(X_\ell) + \sum_{z=j+1}^{\ell} t_z(X_{z-1})$ is exactly the welfare of the matching that gives X_{z-1} to t_z for all $z \in \{j+1,\dots,\ell\}$, and gives X_ℓ to t_j. In other words, both sums represent the welfare of a matching between allocations in $\{X_j,\dots,X_\ell\}$ and types in $\{t_j,\dots,t_\ell\}$. Furthermore, cyclic monotonicity guarantees that the first sum is larger. Therefore, this term is positive, and t_j cannot gain by misreporting any t_ℓ for $\ell > j$. Putting both cases together proves that these prices yield a BIC, IR mechanism, and therefore the bound in Equation (8.1) is attained.

□

8.2 Relating SADP to BMeD

Proposition 12 tells us how, when given an allocation rule that is compatible with the type space, to find prices that achieve the maximum revenue. However, if our goal is to maximize revenue over all feasible allocation rules, it does not tell us what the optimal allocation rule is. Now, let us take a closer look at the bound in Equation (8.1). Each allocation X_ℓ is only ever evaluated by two types: t_ℓ and $t_{\ell+1}$. So to maximize revenue, a tempting approach is to choose X^*_ℓ to be the allocation that maximizes $(\sum_{j\geq \ell} q_j)t_\ell(X_\ell)-(\sum_{j\geq \ell+1} q_j)t_{\ell+1}(X_\ell)$, and hope that that $\vec{X}^*$ and $\vec{t}$ are compatible. Because we chose the X^*_ℓ to maximize the upper bound of Equation (8.1), the optimal obtainable revenue for any allocation rule can not possibly exceed the upper bound of Equation (8.1) when evaluated at the X^*_ℓ (by Proposition 11), and if $\vec{X}^*$ is compatible with $\vec{t}$, then Proposition 12 tells us that this revenue is in fact attainable. Keeping these results in mind, we now begin drawing connections to SADP. For simplicity of notation in the definitions below, we let $f_{n+1}(\cdot)$ be the 0 function.

Definition 6. *(C-compatible) We say that $(f_1,\dots,f_n)$ are C-*compatible *iff there exist multipliers $1 = Q_1 < Q_2 < \dots < Q_n$, of bit complexity at most C, and allocations $(X^*_1,\dots,X^*_n)$ such that X^*_ℓ maximizes $f_\ell(\cdot) - f_{\ell+1}(\cdot)$ for all ℓ and $(X^*_1,\dots,X^*_n)$ is compatible with with $(Q_1f_1,\dots,Q_nf_n)$.*

Observation 11. *Let $1 = Q_1 < Q_2 < \dots < Q_n$. Then*

$$Rev((Q_1f_1,\dots,Q_nf_n),(1/Q_1-1/Q_2,1/Q_2-1/Q_3,\dots,1/Q_n),\vec{X}) \leq \sum_{\ell=1}^{n} f_\ell(X_\ell)-f_{\ell+1}(X_\ell)$$

This bound is tight when $\vec{X}$ is compatible with $(Q_1f_1,\dots,Q_nf_n)$.

Proof. Plug in to Propositions 11 and 12. □

Definition 7. *(D-balanced) For a list of functions $(f_1,\dots,f_n)$, let X^*_ℓ denote the allocation that maximizes $f_\ell(\cdot) - f_{\ell+1}(\cdot)$ for all $\ell \in [n]$. We say that $(f_1,\dots,f_n)$ are D-*balanced *if $f_n(X^*_n) \leq D(f_\ell(X^*_\ell) - f_{\ell+1}(X^*_\ell))$ for all $\ell \in [n-1]$.*

Proposition 13. *For a C-compatible and D-balanced list of functions $(f_1,\dots,f_n)$, let X^*_ℓ denote the allocation that maximizes $f_\ell(\cdot) - f_{\ell+1}(\cdot)$ for all $\ell \in [n]$ and let $1 = Q_1 < Q_2 < \dots < Q_n$ be multipliers such that $(X^*_1,\dots,X^*_n)$ is compatible with $(Q_1f_1,\dots,Q_nf_n)$. If $(X_1,\dots,X_n)$ are such that*

$$\begin{aligned}&Rev((Q_1f_1,\dots,Q_nf_n),(1/Q_1-1/Q_2,1/Q_2-1/Q_3,\dots,1/Q_n),\vec{X})\\ &\qquad\geq \alpha Rev((Q_1f_1,\dots,Q_nf_n),(1/Q_1-1/Q_2,1/Q_2-1/Q_3,\dots,1/Q_n),\vec{X}^*),\end{aligned}$$

then at least one of $\{X_1, \ldots, X_n\}$ *is an* $\left(\alpha - \frac{(1-\alpha)D}{n-1}\right)$*-approximation*[2] *to the SADP instance* $(f_1, \ldots, f_n)$.

Proof. Using Observation 11, we obtain the following chain of inequalities:

$$\begin{aligned}\sum_{\ell=1}^{n} f_\ell(X_\ell) - f_{\ell+1}(X_\ell) &\geq Rev((Q_1 f_1, \ldots, Q_n f_n), (1/Q_1 - 1/Q_2, 1/Q_2 - 1/Q_3, \ldots, 1/Q_n), \vec{X}) \\ &\geq \alpha Rev((Q_1 f_1, \ldots, Q_n f_n), (1/Q_1 - 1/Q_2, 1/Q_2 - 1/Q_3, \ldots, 1/Q_n), \vec{X}^*) \\ &= \alpha \sum_{\ell=1}^{n} (f_\ell(X_\ell^*) - f_{\ell+1}(X_\ell^*))\end{aligned}$$

Rearranging, we can get:

$$\begin{aligned}\sum_{\ell=1}^{n-1} f_\ell(X_\ell) - f_{\ell+1}(X_\ell) &\geq \alpha \left(\sum_{\ell=1}^{n-1} f_\ell(X_\ell^*) - f_{\ell+1}(X_\ell^*) \right) + \alpha f_n(X_\ell^*) - f_n(X_\ell) \\ &\geq \alpha \left(\sum_{\ell=1}^{n-1} f_\ell(X_\ell^*) - f_{\ell+1}(X_\ell^*) \right) - (1-\alpha) f_n(X_\ell^*)\end{aligned}$$

Now, because $(f_1, \ldots, f_n)$ is D-balanced, we have $f_n(X_n^*) \leq D(f_\ell(X_\ell^*) - f_{\ell+1}(X_\ell^*))$ for all ℓ, and therefore $f_n(X_n^*) \leq \frac{D}{n-1} \left(\sum_{\ell=1}^{n-1} f_\ell(X_\ell^*) - f_{\ell+1}(X_\ell^*) \right)$. Using this, we can again rewrite to obtain:

$$\sum_{\ell=1}^{n-1} f_\ell(X_\ell) - f_{\ell+1}(X_\ell) \geq \left(\alpha - \frac{(1-\alpha)D}{n-1} \right) \left(\sum_{\ell=1}^{n-1} f_\ell(X_\ell^*) - f_{\ell+1}(X_\ell^*) \right) \tag{8.2}$$

As choosing the null allocation is always allowed, $f_\ell(X_\ell^*) - f_{\ell+1}(X_\ell^*) \geq 0$ for all $\ell \in [n]$. Now it is easy to see that in order for Equation (8.2) to hold, there must be at least one ℓ with $f_\ell(X_\ell) - f_{\ell+1}(X_\ell) \geq \left(\alpha - \frac{(1-\alpha)D}{n-1}\right)(f_\ell(X_\ell^*) - f_{\ell+1}(X_\ell^*))$. Such an X_ℓ is clearly an $\left(\alpha - \frac{(1-\alpha)D}{n-1}\right)$-approximation to the desired SADP instance. □

With Proposition 13, we are now ready to state our reduction from SADP to BMeD.

Theorem 18. *Let A be an* α*-approximation algorithm for BMeD(*$\mathcal{F}$*,*$\mathcal{V}^\times$*).*[3] *Then an approximate solution to any C-compatible instance* $(f_1, \ldots, f_n)$ *of SADP(*$\mathcal{F}$*,*$\mathcal{V}$*) can be found in polynomial time*

[2] $\alpha - \frac{(1-\alpha)D}{n-1} = 1$ when $\alpha = 1$.

[3] Note that if $\mathcal{V}$ is closed under addition and scalar multiplication then $\mathcal{V}^\times = \mathcal{V}$.

plus one black-box call to A. The solution has the following properties:

1. *(Quality) If $(f_1, \ldots, f_n)$ is D-balanced, the solution obtains an $\left(\alpha - \frac{(1-\alpha)D}{n-1}\right)$-approximation.*

2. *(Bit Complexity) If b is an upper bound on the bit complexity of $f_j(X)$, then $b + C$ is an upper bound on the bit complexity of $t_j(X)$ for any t_j input to A and any $X \in \mathcal{F}$, and every probability input to A has bit complexity C.*

Theorem 18 shows that there is a complete circle of reductions from BMeD to SADP and back. In fact, the circle of exact reductions is complete even if we restrict SADP to instances with $n = 2$. The reduction from SADP to BMeD is not quite approximation preserving, but it is strong enough for us to show hardness of approximation for submodular agents in the following section.

8.3 Approximation Hardness for Submodular Bidders

We begin this section by defining submodularity. There are several known equivalent definitions for submodularity. The one that will be most convenient for us is known as the property of diminishing returns.

Definition 8. *(Submodular Function) A function $f : 2^S \rightarrow \mathbb{R}$ is said to be submodular if for all $X \subset Y \subset S$, and all $x \notin Y$, we have:*

$$f(X \cup \{x\}) - f(X) \geq f(Y \cup \{x\}) - f(Y)$$

Definition 9. *(Value Oracle) A* value oracle *is a black box that takes as input a set S and outputs $f(S)$.*

Definition 10. *(Demand Oracle) A* demand oracle *is a black box that takes as input a vector of prices $p_1, \ldots, p_m$ and outputs* $\text{argmax}_{S \subseteq [m]}\{f(S) - \sum_{i \in S} p_i\}$.

Definition 11. *(Explicit Representation) An* explicit representation *of f is an explicit description of a turing machine that computes f.*

We continue now by providing our hard SADP instance. Let $S_1, \ldots, S_n$ be any subsets of $[m]$ with $|S_i| \leq |S_j|$ for all $i \leq j$. Define $f_i : 2^{[m]} \rightarrow \mathbb{R}$ as the following:

$$f_i(S) = \begin{cases} 2m|S| - |S|^2 & : S \notin \{S_1, \ldots, S_i\} \\ 2m|S| - |S|^2 - 2 + \frac{j}{i} & : S = S_j \ \forall j \in [i] \end{cases}$$

Lemma 16. *Each f_i defined above is a monotone submodular function.*

Proof. Consider any $X \subset Y$, and any $x \notin Y$. Then:

$$\begin{aligned} f_i(X \cup \{x\}) - f_i(X) &\geq 2m(|X| + 1) - (|X| + 1)^2 - 2 - 2m|X| + |X|^2 &&= 2m - 2|X| - 1 \\ f_i(Y \cup \{x\}) - f_i(Y) &\leq 2m(|Y| + 1) - (|Y| + 1)^2 - 2m|Y| + |Y|^2 &&= 2m - 2|Y| + 1 \end{aligned}$$

As $X \subset Y$, we clearly have $|X| < |Y|$, and therefore $f_i(X \cup \{x\}) \geq f_i(Y \cup \{x\})$, so f_i is submodular. Furthermore, as $|X| \leq m$ always, f_i is monotone. □

Proposition 14. *For any $i \in [n-1]$, S_{i+1} maximizes $f_i(\cdot) - f_{i+1}(\cdot)$.*

Proof. For any $S \notin \{S_1, \ldots, S_{i+1}\}$, $f_i(S) - f_{i+1}(S) = 0$. For any $S_j \in \{S_1, \ldots, S_i\}$, $f_i(S_j) - f_{i+1}(S_j) = \frac{j}{i} - \frac{j}{i+1} < 1$. But $f_i(S_{i+1}) - f_{i+1}(S_{i+1}) = 1$, so S_{i+1} maximizes $f_i(\cdot) - f_{i+1}(\cdot)$. □

The idea is in order to solve the SADP instance $(f_1, \ldots, f_n)$ approximately, we need to find one of the S_is with non-neglible probability. Depending on how each f_i is given as input, this may either be easy (if it is given in the form specified above), impossible (if it is given as a value or demand oracle), or computationally hard (if it is given via an arbitrary explicit representation). Impossibility for value and demand oracles is straight-forward, and proved next. The computational hardness result requires only a little extra work.

Lemma 17. *Let $\mathcal{S} = \{S_1, \ldots, S_a\}$ be subsets chosen uniformly at random (without replacement) from all subsets of $[m]$, then listed in increasing order based on their size. Then no algorithm exists that can guarantee an output of $S \in \mathcal{S}$ with probability $\frac{a(c+1)}{2^m - c}$ given only c value or demand oracle queries of $f_1, \ldots, f_n$.*

Proof. For any sequence of c value queries, the probability that none of them guessed a set $S \in \mathcal{S}$ is at least $1 - ac/(2^m - c)$. For a deterministic algorithm using c value queries, if it has never successfully queried an element in $\mathcal{S}$ during the execution, then the probability of success is no more than $a/(2^m - c)$. Using union bound, we know that the probability of success for any deterministic algorithm is at most $a(c+1)/(2^m - c)$. By Yao's Lemma, this also holds for randomized algorithms.

A similar argument shows that the same holds for demand queries as well. First, let's understand how demand queries for f_0 ($f_0(S) = 2m|S| - |S|^2$) and f_i differ. Consider any prices $p_1, \dots, p_m$, and let $S = \text{argmax}\{f_0(S) - \sum_{i \in S} p_i\}$. If $S \notin \mathcal{S}$, then because $f_a(S) = f_0(S)$ and $f_a(S') \leq f_0(S')$ for all S', we also have $S = \text{argmax}\{f_a(S) - \sum_{i \in S} p_i\}$. If other words, if the prices we query are such that a demand oracle for f_0 would return some $S \notin \mathcal{S}$, we have learned nothing about $\mathcal{S}$ other than that it does not contain S. From here we can make the exact same argument in the previous paragraph replacing the phrase "query some $S \in \mathcal{S}$" with "query some prices $p_1, \dots, p_m$ such that a demand oracle for f_0 would return some $S \in \mathcal{S}$." □

Corollary 9. *For any constants c, d, no (possibly randomized) algorithm can guarantee a $\frac{1}{m^c}$-approximation to SADP($2^{[m]}$,monotone submodular functions) in time $(nm)^d$, as long as $n \leq m^{c'}$ for some constant c', if the functions are given via a value or demand oracle.*

Proof. Consider the input $f_1, \dots, f_n$ chosen according to Lemma 17. Therefore by Proposition 14, in any $\frac{1}{m^c}$-approximation to this SADP instance necessarily finds some S_i with probability at least $\frac{1}{m^c}$. Lemma 17 exactly says that this cannot be done without at least $2^m/n >> (nm)^d$ queries. □

Corollary 9 shows that SADP is impossible to approximate within any polynomial factor for submodular functions given via a value or demand oracle. We continue now by showing computational hardness when the functions are given via an explicit representation. First, let g be any correspondence between subsets of $[m]$ and integers in $[2^m]$ so that $|S| > |S'| \Rightarrow g(S) > g(S')$.[4] Next, let $\mathcal{P}$ denote any NP-hard problem with verifier V, and let p be a constant so that any "yes" instance of $\mathcal{P}$ of size z has a bitstring witness of length z^p. For a given m, n, and specific instance $P \in \mathcal{P}$ of size $(m - \log n)^{1/p}$, we define:

[4]Such a g can be implemented efficiently by ordering subsets of $[m]$ first by their size, and then lexicographically, and letting $g(S)$ be the rank of S in this ordering. The rank of S in this ordering can be computed by easy combinatorics plus a simple dynamic program.

$$f_i^P(S) = \begin{cases} 2m|S| - |S|^2 - 2 + \frac{j}{i} & : \frac{(j-1)2^m}{n} < g(S) \le \frac{j2^m}{n} \le \frac{i2^m}{n} \text{ and } V\left(P, (g(S) \mod \frac{2^m}{n})\right) = \text{“}yes\text{''} \\ 2m|S| - |S|^2 & : \text{otherwise} \end{cases}$$

Lemma 18. *If P is a “yes” instance of $\mathcal{P}$, then any α-approximation to the SADP instance $f_1, \ldots, f_n$ necessarily finds a witness for P with probability at least α.*

Proof. It is clear that if P is a “yes” instance of $\mathcal{P}$, then P has a witness of length $(m - \log n)$, and therefore there is some $x \le 2^m/n$ such that $V(P, x) = \text{“}yes.\text{''}$ Therefore, we would have $f_i^P(g^{-1}(x + i2^m/n)) = 2m|S| - |S|^2$, but $f_{i+1}^P(g^{-1}(x + i2^m/n)) = 2m|S| - |S|^2 - 1$. It is also clear that in order to have $f_i^P(S) - f_{i+1}^P(S) > 0$, we must have $V(P, (g(S) \mod 2^m/n)) = \text{“}yes\text{''}$. In other words, if P is a “yes” instance of $\mathcal{P}$, any α-approximation to the SADP instance $f_1, \ldots, f_n$ must find a witness for P with probability at least α. □

Corollary 10. *Unless NP = RP, for any constants c, d no (possibly randomized) algorithm can guarantee a $\frac{1}{m^c}$-approximation to SADP($2^{[m]}$,monotone submodular functions) in time $(nm)^d$, as long as $n \le m^{c'}$ for some constant c', even if the functions are given explicitly.*

Proof. If we could obtain a $\frac{1}{m^c}$-approximation to SADP($2^{[m]}$,monotone submodular functions) in time $(nm)^d \le m^{c'd+d}$, then for any “yes” instance $P \in \mathcal{P}$, we could find a witness with probability at least $\frac{1}{m^c}$ in time $m^{c'd+d}$ by Lemma 18. By running m^c independent trials, we could amplify this probability to $1/e$ in time $m^{c'd+d+c}$. So we would have an *RP* algorithm for $\mathcal{P}$. □

We combine Corollaries 9 and 10 below into one theorem.

Theorem 19. *As long as $n \le m^{c'}$ for some constant c', the problem SADP($2^{[m]}$,monotone submodular functions) is:*

1. *Impossible to approximate within any* $1/\text{poly}(m)$*-factor with only* $\text{poly}(n, m)$ *value oracle queries.*

2. *Impossible to approximate within any* $1/\text{poly}(m)$*-factor with only* $\text{poly}(n, m)$ *demand oracle queries.*

3. *Impossible to approximate within any* $1/\text{poly}(m)$*-factor given explicit access to the input functions in time* $\text{poly}(n, m)$*, unless* $NP = RP$*.*

Finally, we conclude this section by showing that the classes which we have just shown to be hard for SADP can be used via the reduction of Theorem 18.

Lemma 19. *All SADP instances defined in this section are* $2n(\log n + \log m)$*-compatible and* m^2*-balanced.*

Proof. That each instance is m^2-balanced is easy to see: $f_n([m]) = m^2$, and for all i we have $\max_S\{f_i(S) - f_{i+1}(S)\} = 1$. To see that each instance is C-compatible, set $Q_i = (nm)^{2i-2}$. Then it is easy to see that the maximum welfare obtainable by any matching that only uses types $Q_1 f_1$ through $Q_i f_i$ obtains welfare at most $m^2 i Q_i$. In addition, the difference in value of $Q_{i+1} f_{i+1}$ between two outcomes is at least $Q_{i+1}/(i+1) > Q_{i+1}/n$ if it is non-zero. Therefore, the minimum non-zero difference between the value of $Q_{i+1} f_{i+1}$ for two allocations is larger than the maximum possible difference in welfare of all types $Q_1 f_1$ through $Q_i f_i$ for two matchings of allocations. As this holds for all i, this means that the max-weight matching of any set of allocations S to types $\{Q_i f_i, \dots, Q_j f_j\}$ will necessarily match $Q_j f_j$ to its favorite allocation in S, then $Q_{j-1} f_{j-1}$ to its favorite of what remains, etc.

So let $(S_1, \dots, S_n)$ be any sets such that $f_i(S_{i+1}) - f_i(S_i) = 1$. It is not hard to verify that $|S_1| \leq |S_2| \leq \dots \leq |S_n|$, and $f_i(S_j)$ is monotonically increasing in j for any i. The reasoning above immediately yields that the max-weight matching of allocations $(S_i, \dots, S_j)$ to types $(Q_i f_i, \dots, Q_j f_j)$ necessarily awards S_ℓ to $Q_\ell f_\ell$ for all ℓ, as f_i, so $(S_1, \dots, S_n)$ is cyclic monotone with respect to $(Q_1 S_1, \dots, Q_n S_n)$. Furthermore, the same reasoning immediately yields that the max-weight matching of allocations $(S_i, \dots, S_{j-1})$ to types $(Q_{i+1} f_{i+1}, \dots, Q_j f_j)$ awards S_ℓ to type $Q_{\ell+1} f_{\ell+1}$ for all ℓ, so $(S_1, \dots, S_n)$ is also compatible with $(Q_1 f_1, \dots, Q_n f_n)$. It is clear that all Q_i are integers less than $(n^2 m^2)^{n-1} \leq 2^{2n(\log n + \log m)}$, so $(f_1, \dots, f_n)$ are $2n(\log n + \log m)$-compatible. □

Corollary 11. *The problem BMeD(*$2^{[m]}$*,monotone submodular functions) (for* $n = |T| = \text{poly}(m)$*) is:*

1. *Impossible to approximate within any* $1/\text{poly}(m)$*-factor with only* $\text{poly}(n, m)$ *value oracle queries.*

2. *Impossible to approximate within any* $1/\text{poly}(m)$*-factor with only* $\text{poly}(n, m)$ *demand oracle queries.*

3. *Impossible to approximate within any* $1/\text{poly}(m)$*-factor given explicit access to the input functions in time* $\text{poly}(n, m)$*, unless* $NP = RP$*.*

Proof. For any constant c, let $n = m^{c+2} + 1$, if we can find an $2/m^c$-approximate solution to BMeD in polynomial time, we can find an $1/m^c$-approximate solution to SADP by Theorem 18, which would contradict Theorem 19. □

Chapter 9

Conclusions and Open Problems

As mediums like the Internet continue to grow, we see more and more that our algorithms are interacting with strategic agents. These agents often have incentives to manipulate the algorithms for their own gains, resulting in potentially global failures of otherwise sound algorithms [McC08, McM08]. Such instances demonstrate the necessity of algorithms that are robust to potential manipulation by strategic agents, called mechanisms. In this thesis, we addressed a fundamental question: How much more difficult is mechanism design than algorithm design?

In this study, we addressed two important areas of research. The first follows Myerson's seminal paper on optimal auction design, where he showed that the optimal single item auction is a virtual welfare maximizer [Mye81]. Much effort had been devoted over the past three decades to extend his characterization to multi-dimensional settings, but no compelling characterization in unrestricted settings had been found. Our work directly extends Myerson's characterization to unrestricted multi-dimensional settings, showing that the optimal auction is a distribution over virtual welfare maximizers.

This thesis also follows Nisan and Ronen's seminal paper introducing the field of Algorithmic Mechanism Design [NR99]. On this front, our work provides a black-box reduction from mechanism to algorithm design. The existence of this reduction *does not* imply that we are now capable of solving any mechanism design question asked, but it *does* provide an algorithmic framework with which to tackle such problems, which was previously non-existent. We further make use of our new framework to design the first computationally efficient, approximately-optimal mechanisms for the paradigmatic problems of makespan minimization and fairness maximization in unrestricted settings. Furthermore, our framework allows for the first unrestricted hardness of approximation result in revenue maximization.

On both fronts, our work constitutes some immense progress, but there is still much work to be done. Myerson's single-dimensional characterization combined with results of Bulow and Roberts provides a meaningful economic interpretation of the transformation from types to virtual types [Mye81, BR89], whereas our virtual transformation is randomized and computed by a linear program. While the use of randomization is necessary in even very simple multi-dimensional settings [BCKW10, DDT13, HN13, HR12, Pav06, Tha04], it is still conceivable that these randomized virtual transformations have economic significance, at least in some special cases.

Question 7. *Is there any economic significance to the virtual transformations used by our mechanisms?*

We have also established a "tight" algorithmic framework to design multi-dimensional mechanisms, namely by designing algorithms for virtual welfare maximization. Applied to the case of additive agents with feasibility constraints on which agents can simultaneously receive which items (e.g. matching markets), we obtain numerous positive results (e.g. Theorems 4, 8, 9). Applied to the case of a single agent whose value for sets of items is submodular, we obtain hardness of approximation within any polynomial factor (Theorem 19 and Corollary 11). Still, there are numerous interesting classes of valuation functions between additive and submodular worthy of study in multi-dimensional mechanism design, and this space of questions is largely unexplored. Our framework proposes concrete algorithmic questions for use in this study.

Question 8. *What is the computational complexity of virtual welfare maximization (and therefore truthful revenue maximization as well via our reduction) for other classes of valuation functions? One important instance is the class of* gross substitute *valuations.*

A final question motivated in our work in multi-dimensional mechanism is whether or not our mechanisms are prescriptive for the sale of multiple items in the real world. Most multi-item sellers still opt to auction items in their inventory separately, even though numerous examples show that this could potentially hurt revenue by a significant factor. One possibility indeed is that sellers should consider using more complex mechanisms, and another is that "real-world" instances don't behave as poorly as the worst case. Understanding which world we live in is of obvious importance, and our mechanisms provide a computational tool with which to address this question (i.e. by executing our mechanisms on real-world instances).

Question 9. *What is the gap between simple and optimal mechanisms in "real-world" instances? If the gap is large, should sellers change the way they sell items? If the gap is small, what properties of real-world instances enable this?*

Within algorithmic mechanism design, our framework poses new concrete algorithmic questions. If one wants to design mechanisms optimizing any objective, one can instead use our framework and design algorithms for that same objective plus virtual welfare. Algorithm design questions like these are largely unstudied, as they were previously unmotivated. In light of our framework, it is now quite important to solve such problems.

Question 10. *For what other optimization problems $\mathcal{O}$ can we design computationally efficient algorithms for $\mathcal{O}+$ virtual welfare?*

Beyond the design of computationally efficient mechanisms, an equally pressing direction is understanding the computational complexity of such mechanism design, e.g. by proving hardness of approximation. Given the success of our framework in this direction for multi-dimensional mechanism design, we are optimistic that our framework can help provide such results in algorithmic mechanism design as well. Of course, extending our framework to accommodate reductions from algorithm to mechanism design was already challenging in the case of multi-dimensional mechanism design, and will be even more challenging here. Consider for example job scheduling on unrelated machines: we have shown that there is a computationally efficient 2-approximate mechanism via a $(1, 1/2)$-approximation algorithm for makespan minimization with costs, yet makespan minimization with costs is NP-hard to approximate (in a traditional sense) within any finite factor. In other words, any such reduction from algorithm to mechanism design must somehow accommodate this fact and make use of (α,β)-approximations as well.

Question 11. *Can one extend our framework for algorithmic mechanism design to allow for the development of hardness of approximation results as well?*

Finally, recall that there are two central questions in the field of algorithmic mechanism design, refining our motivating question of "how much harder is mechanism than algorithm design?" This thesis makes substantial progress in addressing the computational question: How much *computationally* harder is mechanism than algorithm design? We show that the answer for makespan and fairness is "not at all," and provide a framework with which to answer this question for other objectives. But another important question simply addresses the quality of solution output by the optimal mechanism versus the optimal algorithm, without regard for computation. Our work does not directly address this question, but by designing computationally efficient mechanisms, we provide a new algorithmic tool with which to make progress.

Question 12. *How much worse is the quality of solution output by the optimal mechanism versus that of the optimal algorithm?*

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.

[AFHH13] Saeed Alaei, Hu Fu, Nima Haghpanah, and Jason Hartline. The Simple Economics of Approximately Optimal Auctions. In *Proceedings of the 54th IEEE Symposium on Foundations of Computer Science (FOCS)*, 2013.

[AFS08] Arash Asadpour, Uriel Feige, and Amin Saberi. Santa claus meets hypergraph matchings. In *the 12th International Workshop on Approximation, Randomization, and Combinatorial Optimization (APPROX-RANDOM)*, 2008.

[AKW14] Pablo D. Azar, Robert Kleinberg, and S. Matthew Weinberg. Prophet inequalities with limited information. In *the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2014.

[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, 1996.

[AS07] Arash Asadpour and Amin Saberi. An approximation algorithm for max-min fair allocation of indivisible goods. In *the 39th Annual ACM Symposium on Theory of Computing (STOC)*, 2007.

[AT01] Aaron Archer and Éva Tardos. Truthful Mechanisms for One-Parameter Agents. In *the 42nd Annual Symposium on Foundations of Computer Science (FOCS)*, 2001.

[BCG09] MohammadHossein Bateni, Moses Charikar, and Venkatesan Guruswami. Maxmin allocation via degree lower-bounded arborescences. In *the 41st annual ACM symposium on Theory of Computing (STOC)*, pages 543–552, 2009.

[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.

[BD05] Ivona Bezáková and Varsha Dani. Allocating indivisible goods. *SIGecom Exchanges*, 5(3):11–18, 2005.

[BDF+10] David Buchfuhrer, Shaddin Dughmi, Hu Fu, Robert Kleinberg, Elchanan Mossel, Christos H. Papadimitriou, Michael Schapira, Yaron Singer, and Christopher Umans. Inapproximability for VCG-Based Combinatorial Auctions. In *Proceedings of 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.

[BGM13] Anand Bhalgat, Sreenivas Gollapudi, and Kamesh Munagala. Optimal Auctions via the Multiplicative Weight Method. In *the 14th ACM Conference on Electronic Commerce (EC)*, 2013.

[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.

[BHSZ13] MohammadHossein Bateni, Nima Haghpanah, Balasubramanian Sivan, and Morteza Zadimoghaddam. Revenue maximization with nonexcludable goods. In *the 9th International Conference on Web and Internet Economics (WINE)*, 2013.

[BILW14] Moshe Babaioff, Nicole Immorlica, Brendan Lucier, and S. Matthew Weinberg. A simple and approximately optimal mechanism for an additive buyer. *Working Paper*, 2014. http://people.csail.mit.edu/smweinberg/additive.pdf.

[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.

[BR89] J. Bulow and J. Roberts. The Simple Economics of Optimal Auctions. *Journal of Political Economy*, 97:1060–1090, 1989.

[Bri08] Patrick Briest. Uniform Budgets and the Envy-Free Pricing Problem. In *Proc. of the 35th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 808–819, 2008.

[BS06] Nikhil Bansal and Maxim Sviridenko. The santa claus problem. In *the 38th annual ACM symposium on Theory of computing (STOC)*, pages 31–40, 2006.

[BT96] Steven J Brams and Alan D Taylor. *Fair Division: From cake-cutting to dispute resolution*. Cambridge University Press, 1996.

[Bud13] Zack Budryk. Dangerous curves. *Inside Higher Ed*, 2013. http://www.insidehighered.com/news/2013/02/12/students-boycott-final-challenge-professors-grading-policy-and-get.

[CCK09] Deeparnab Chakrabarty, Julia Chuzhoy, and Sanjeev Khanna. On allocating goods to maximize fairness. In *50th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2009.

[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.

[CDP+14] Xi Chen, Ilias Diakonikolas, Dimitris Paparas, Xiaorui Sun, and Mihalis Yannakakis. The complexity of optimal multidimensional pricing. In *the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2014.

[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. http://arxiv.org/pdf/1305.4000v1.pdf.

[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. http://arxiv.org/pdf/1305.4002v1.pdf.

[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.

[CK10] George Christodoulou and Annamária Kovács. A Deterministic Truthful PTAS for Scheduling Related Machines. In *the 21st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2010.

[CKK07] George Christodoulou, Elias Koutsoupias, and Annamária Kovács. Mechanism Design for Fractional Scheduling on Unrelated Machines. In *the 34th International Colloquium on Automata, Languages and Programming (ICALP)*, 2007.

[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 *the 18th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2007.

[Cla71] Edward H. Clarke. Multipart pricing of public goods. *Public Choice*, pages 17–33, 1971.

[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.

[CP14] Yang Cai and Christos Papadimitriou. Simultaneous bayesian auctions and computational complexity. In *Proceedings of the 15th ACM Conference on Electronic Commerce (EC)*, 2014.

[CS08] Vincent Conitzer and Tuomas Sandholm. New Complexity Results about Nash Equilibria. *Games and Economic Behavior*, 63(2):621–641, 2008.

[DDDR08] Peerapong Dhangwatnotai, Shahar Dobzinski, Shaddin Dughmi, and Tim Roughgarden. Truthful Approximation Schemes for Single-Parameter Agents. In *the 49th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2008.

[DDT13] Constantinos Daskalakis, Alan Deckelbaum, and Christos Tzamos. Mechanism design via optimal transport. In *the 14th ACM Conference on Electronic Commerce (EC)*, 2013.

[DDT14] Constantinos Daskalakis, Alan Deckelbaum, and Christos Tzamos. The complexity of optimal mechanism design. In *the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2014.

[DFK11] Shahar Dobzinski, Hu Fu, and Robert D. Kleinberg. Optimal Auctions with Correlated Bidders are Easy. In *the 43rd ACM Symposium on Theory of Computing (STOC)*, 2011.

[DGP09] Constantinos Daskalakis, Paul W. Goldberg, and Christos H. Papadimitriou. The Complexity of Computing a Nash Equilibrium. *SIAM Journal on Computing*, 39(1):195–259, 2009.

[DJ81] E. Davis and J.M. Jaffe. Algorithms for scheduling tasks on unrelated processors. *Journal of the Association for Computing Machinery*, 28:821–736, 1981.

[Dob11] Shahar Dobzinski. An Impossibility Result for Truthful Combinatorial Auctions with Submodular Valuations. In *Proceedings of the 43rd ACM Symposium on Theory of Computing (STOC)*, 2011.

[DV12] Shahar Dobzinski and Jan Vondrak. The Computational Complexity of Truthfulness in Combinatorial Auctions. In *Proceedings of the ACM Conference on Electronic Commerce (EC)*, 2012.

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

[DW14] Constantinos Daskalakis and S. Matthew Weinberg. Bayesian truthful mechanisms for job scheduling from bi-criterion approximation algorithms. *Working Paper*, 2014. http://people.csail.mit.edu/smweinberg/makespan.pdf.

[GIS77] T. Gonzalez, O.H. Ibarra, and S. Sahni. Bounds for lpt schedules on uniform processors. *SIAM Journal on Computing*, 6:155–166, 1977.

[GJ75] M.R. Garey and D.S. Johnson. Complexity results for multiprocessor scheduling under resource constraints. *SIAM Journal on Computing*, 4:397–411, 1975.

[GJ78] M.R. Garey and D.S. Johnson. Strong np-completeness results: Motivation, examples, and implications. *Journal of the Association for Computing Machinery*, 25:499–508, 1978.

[GLLK79] R.L. Graham, E.L. Lawler, J.K. Lenstra, and A.H.G. Rinooy Kan. Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of Discrete Mathematics*, 5:287–326, 1979.

[GLS81] Martin Grötschel, László Lovász, and Alexander Schrijver. The Ellipsoid Method and its Consequences in Combinatorial Optimization. *Combinatorica*, 1(2):169–197, 1981.

[Gra66] R.L. Graham. Bounds for certain multiprocessing anomalies. *Bell System Technological Journal*, 45:1563–1581, 1966.

[Gra69] R.L. Graham. Bounds on multiprocessing timing anomalies. *SIAM Journal on Applied Mathematics*, 17:416–429, 1969.

[Gro73] Theodore Groves. Incentives in teams. *Econometrica*, pages 617–623, 1973.

[HIMM11] Nima Haghpanah, Nicole Immorlica, Vahab S. Mirrokni, and Kamesh Munagala. Optimal auctions with positive network externalities. In *the 12th ACM Conference on Electronic Commerce (EC)*, 2011.

[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.

[HN13] Sergiu Hart and Noam Nisan. The Menu-Size Complexity of Auctions. In *the 14th ACM Conference on Electronic Commerce (EC)*, 2013.

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

[HR09] J.D. Hartline and T. Roughgarden. Simple versus optimal mechanisms. In *Proceedings of the 10th ACM conference on Electronic commerce*, pages 225–234. ACM, 2009.

[HR11] Sergiu Hart and Philip J. Reny. Implementation of reduced form mechanisms: A simple approach and a new characterization. *Technical Report, The Hebrew University of Jerusalem, Center for Rationality DP-594*, 2011. http://www.ma.huji.ac.il/hart/abs/q-mech.html.

[HR12] Sergiu Hart and Philip J. Reny. Maximal revenue with multiple goods: Nonmonotonicity and other obsevations. *Technical Report, The Hebrew University of Jerusalem, Center for Rationality DP-630*, 2012. http://www.ma.huji.ac.il/hart/abs/monot-m.html.

[HS76] E. Horowitz and S. Sahni. Exact and approximate algorithms for scheduling nonidentical processors. *Journal of the Association for Computing Machinery*, 23:317–327, 1976.

[HS87] D.S. Hochbaum and D.B. Shmoys. Using dual approximation algorithms for scheduling problems: practical and theoretical results. *Journal of the Association for Computing Machinery*, 34:144–162, 1987.

[HS88] D.S. Hochbaum and D.B. Shmoys. A polynomial approximation scheme for machine scheduling on uniform processors: using the dual approximation approach. *SIAM Journal on Computing*, 17:539–551, 1988.

[Kha79] Leonid G. Khachiyan. A Polynomial Algorithm in Linear Programming. *Soviet Mathematics Doklady*, 20(1):191–194, 1979.

[Kna46] Bronislaw Knaster. Sur le probleme du partage pragmatique de h. steinhaus. In *Annales de la Societé Polonaise de Mathematique*, volume 19, pages 228–230, 1946.

[KP80] Richard M. Karp and Christos H. Papadimitriou. On Linear Characterizations of Combinatorial Optimization Problems. In *the 21st Annual Symposium on Foundations of Computer Science (FOCS)*, 1980.

[KP07] Subhash Khot and Ashok Kumar Ponnuswami. Approximation algorithms for the max-min allocation problem. In *the 11th International Workshop on Approximation, Randomization, and Combinatorial Optimization (APPROX-RANDOM)*, 2007.

[KV07] Elias Koutsoupias and Angelina Vidali. A Lower Bound of $1+\phi$ for Truthful Scheduling Mechanisms. In *the 32nd International Symposium on the Mathematical Foundations of Computer Science (MFCS)*, 2007.

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

[LST87] Jan Karel Lenstra, David B. Shmoys, and Éva Tardos. Approximation algorithms for scheduling unrelated parallel machines. In *FOCS*, 1987.

[Lu09] Pinyan Lu. On 2-Player Randomized Mechanisms for Scheduling. In *the 5th International Workshop on Internet and Network Economics (WINE)*, 2009.

[LY08a] Pinyan Lu and Changyuan Yu. An Improved Randomized Truthful Mechanism for Scheduling Unrelated Machines. In *the 25th Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, 2008.

[LY08b] Pinyan Lu and Changyuan Yu. Randomized Truthful Mechanisms for Scheduling Unrelated Machines. In *the 4th International Workshop on Internet and Network Economics (WINE)*, 2008.

[LY13] Xinye Li and Andrew Chi-Chih Yao. On revenue maximization for selling multiple independently distributed items. *Proceedings of the National Academy of Sciences*, 110(28):11232–11237, 2013.

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

[McC08] Declan McCullagh. How Pakistan knocked YouTube offline (and how to make sure it never happens again). *CNET*, 2008. http://news.cnet.com/8301-10784_3-9878655-7.html.

[McM08] Robert McMillan. Weekend youtube outage underscores big internet problem. *Macworld*, 2008. http://www.macworld.com/article/1132256/networking.html.

[Mis12] Debasis Mishra. Multidimensional Mechanism Design: Key Results and Research Issues. *Current Science*, 103(9), 2012.

[MM88] Preston McAfee and John McMillan. Multidimensional Incentive Compatibility and Mechanism Design. *Journal of Economic Theory*, 46(2):335–354, 1988.

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

[MS07] Ahuva Mu'alem and Michael Schapira. Setting lower bounds on truthfulness: extended abstract. In *the 18th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2007.

[MV06] Alejandro Manelli and Daniel Vincent. Bundling as an Optimal Selling Mechanism for a Multiple-Good Monopolist. *Journal of Economic Theory*, 127(1):1–35, 2006.

[MV10] A. M. Manelli and D. R. Vincent. Bayesian and Dominant-Strrategy Implementation in the Independent Private-Values Model. *Econometrica*, 78(6):1905–1938, 2010.

[Mye79] Roger B. Myerson. Incentive Compatibility and the Bargaining Problem. *Econometrica*, 41:61–73, 1979.

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

[Nas51] John F. Nash. Non-Cooperative Games. *Annals of Mathematics*, 54(2):286–295, 1951.

[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.

[OR94] Martin J. Osborne and Ariel Rubinstein. *A Course in Game Theory*. MIT Press, 1994.

[Pav06] Gregory Pavlov. Optimal mechanism for selling substitutes, 2006. Boston University - Department of Economics - Working Papers Series WP2006-014.

[Pav11] Gregory Pavlov. A Property of Solutions to Linear Monopoly Problems. *B. E. Journal of Theoretical Economics*, 11(1), 2011. Article 4.

[Pot85] C.N. Potts. Analysis of a linear programming heuristic for scheduling unrelated parallel machines. *Discrete Applied Mathematics*, 10:155–164, 1985.

[PP11] Christos H. Papadimitriou and George Pierrakos. On Optimal Single-Item Auctions. In *the 43rd ACM Symposium on Theory of Computing (STOC)*, 2011.

[PSS08] Christos H. Papadimitriou, Michael Schapira, and Yaron Singer. On the hardness of being truthful. In *Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2008.

[RC98] Jean-Charles Rochet and Philippe Choné. Ironing, Sweeping, and Multidimensional Screening. *Econometrica*, 66(4):783–826, 1998.

[Rob79] Kevin Roberts. The Characterization of Implementable Choice Rules. *Aggregation and Revelation of Preferences*, pages 321–329, 1979.

[Roc87] Jean-Charles Rochet. A Necessary and Sufficient Condition for Rationalizability in a Quasi-Linear Context. *Journal of Mathematical Economics*, 16:191–200, 1987.

[RZ83] John Riley and Richard Zeckhauser. Optimal Selling Strategies: When to Haggle, When to Hold Firm. *Quarterly J. Economics*, 98(2):267–289, 1983.

[Sah76] S. Sahni. Algorithms for scheduling independent tasks. *Journal of the Association for Computing Machinery*, 23:116–127, 1976.

[ST93a] David B Shmoys and Éva Tardos. An approximation algorithm for the generalized assignment problem. *Mathematical Programming*, 62(1-3):461–474, 1993.

[ST93b] David B. Shmoys and Éva Tardos. Scheduling Unrelated Machines with Costs. In *the 4th Symposium on Discrete Algorithms (SODA)*, 1993.

[Ste48] Hugo Steinhaus. The problem of fair division. *Econometrica*, 16(1), 1948.

[Tha04] John Thanassoulis. Haggling Over Substitutes. *J. Economic Theory*, 117:217–245, 2004.

[TW14] Pingzhong Tang and Zihe Wang. Optimal mechanisms with simple menus. In *Proceedings of the 15th ACM Conference on Electronic Commerce (EC)*, 2014.

[Vic61] William Vickrey. Counterspeculation, auctions and competitive sealed tenders. *Journal of Finance*, pages 8–37, 1961.

[Voh11] Rakesh Vohra. *Mechanism Design: A Linear Programming Approach.* Cambridge University Press, 2011.

Appendix A

Extensions of Border's Theorem

In this chapter, we provide two extensions of Border's Theorem for the feasibility of single-item reduced forms. These results further strengthen our characterization of feasible multi-item mechanisms. Despite their importance, we choose to state the results separately here because they do not generalize beyond additive settings. Section A.1 provides an algorithmic extension, showing that we can determine if a reduced form is feasible or not by checking only a linear number of constraints, rather than every possible violating set. Section A.2 provides a stronger structural guarantee on feasible mechanisms.

A.1 Algorithmic Extension

We begin by reviewing Border's Theorem.

Theorem 20 ([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*

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

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.

Recently, an alternative proof of Border's theorem for distributions with finite support was discovered in [Bor07] and again in [CKM11]. These proofs extend Theorem 20 to independent, but not necessarily identical, bidders and non-symmetric reduced forms. In this case, (A.1) is replaced by the following necessary and sufficient condition:

$$\forall S_1 \subseteq T, \dots, S_k \subseteq T : \sum_i \sum_{A \in S_i} \pi_i(A) \Pr[t_i = A] \leq 1 - \prod_i (1 - \Pr[t_i \in S_i]). \tag{A.2}$$

The interpretation of the LHS and RHS of the above inequality is the same as the one given above for (A.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 $S = (S_1, \dots, S_k)$, then there is also a constricting set of the form $S' = (S^{(1)}_{x_1}, \dots, S^{(k)}_{x_k})$, where $S^{(i)}_{x_i} = \{A \in T | \pi_i(A) > x_i\}$. In other words, each bidder has a different threshold x_i, and $S^{(i)}_{x_i}$ contains all types of bidder i with π_i above x_i. Unfortunately, despite this simplification, there are still $(|T| + 1)^k$ 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 $|T|$ and k. We prove such a theorem by introducing a notion of a scaled reduced form, defined next. The idea behind the scaled reduced form is that the most difficult types to satisfy are not those with the large values in the reduced form, but those with large scaled values in the reduced form. We denote the scaled reduced form by $\hat{\pi}$.

Definition 12. *If π is a reduced form, we define its corresponding* scaled reduced form $\hat{\pi}$ *as follows: for all i and types $A \in T$, $\hat{\pi}_i(A) := \Pr[\pi_i(t_i) \leq \pi_i(A)]\pi_i(A)$.*

It turns out that this definition exactly captures which types of different bidders are harder to satisfy. In the bidder-symmetric case Theorem 20, we were able to compare a pair of types A and B

submitted by bidders $i \neq i'$ based only on their corresponding $\pi_i(A)$ and $\pi_{i'}(B)$. 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 $\hat{\pi}_{i'}(B)$:

Theorem 21. *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 $S_x^{(i)} = \{A \in T | \hat{\pi}_i(A) > x\}$ satisfy:*

$$\sum_i \sum_{A \in S_x^{(i)}} \pi_i(A) \Pr[t_i = A] \leq 1 - \prod_i (1 - \Pr[t_i \in S_x^{(i)}]). \tag{A.3}$$

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 $k|T| \cdot \log(k|T|)$.

Before proving Theorem 21, we prove that the concept of a scaled reduced form is necessary.

Proposition 15. *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 15 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 scaled reduced forms. Here is a quick observation about the scaled reduced forms, followed by a proof of Theorem 21.

Observation 12. *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. $\square$

Proof of Theorem 21: 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 $S_{x_i} = \{A \mid \pi_i(A) \geq x_i, A \in T\}$. (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 $A \in T$ carries also the name of the bidder for whom this is their type. In other words, think of each type as an ordered pair (i, A).) Now consider any minimal constricting set of this form, i.e. a choice of $x_1, \ldots, x_k$ 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. [1] Now let $(i, A) \in \operatorname{argmin}_{i,\, A \in S_{x_i}} \hat{\pi}_i(A)$. Then by Observation 12 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 $A' \in T$, $A' \notin S$ and $\hat{\pi}_\ell(A') \geq \hat{\pi}_i(A)$. Observe first that we must have A' from some bidder $\ell \neq i$, as every $A'' \in T$ with $\hat{\pi}_i(A'') \geq \hat{\pi}_i(A)$ has $\pi_i(A'') \geq \pi_i(A) \geq x_i$, so we would

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

have $A'' \in S$. So for this A', we have:

$$\begin{aligned}
&\pi_\ell(A')\Pr[\pi_\ell(t_\ell) \le \pi_\ell(A')] \ge \pi_i(A)\Pr[\pi_i(t_i) \le \pi_i(A)] \\
\Longrightarrow &\pi_\ell(A')\Pr[\pi_\ell(t_\ell) < x_\ell] \ge \pi_i(A)\Pr[\pi_i(t_i) < \pi_i(A)] \\
\Longrightarrow &\pi_\ell(A')\Pr[\pi_\ell(t_\ell) < x_\ell] \ge \pi_i(A)\Pr[\pi_i(t_i) < x_i] \\
\Longrightarrow &\pi_\ell(A')\prod_{j\neq i}\Pr[\pi_j(t_j) < x_j] \ge \pi_i(A)\prod_{j\neq \ell}\Pr[\pi_j(t_j) < x_j] \\
\Longrightarrow &\frac{\pi_k(A')}{\prod_{j\neq\ell}\Pr[\pi_j(t_j) < x_j]} \ge \frac{\pi_i(A)}{\prod_{j\neq i}\Pr[\pi_j(t_j) < x_j]}.
\end{aligned}$$

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

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

$$\Longrightarrow \quad \Pr[t_\ell = A']\pi_\ell(A') > \Pr[t_\ell = A']\prod_{j\neq\ell}\Pr[\pi_j(t_j) < x_j].$$

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 $\Pr[t_\ell = A']\pi_\ell(A')$. And the change to the right-hand side is exactly $\Pr[t_\ell = A']$ times the probability that none of the types in $\cup_{j\neq\ell}S'_j$ show up. As we add more types to S, the probability that none of the types in $\cup_{j\neq\ell}S'_j$ show up will never increase. So for any constricting set S' containing S, we can add A' to S'_ℓ and still get a constricting set.

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

A.2 Structural Extension

Here, we provide a structural extension of Border's Theorem. A corollary of Border's Theorem is that every feasible reduced form can be implemented as a distribution over *hierarchical mechanisms*, that is, mechanisms that maintain a total ordering of all types and award the item to the highest type in that ordering who shows up. Essentially, one proves this by showing that the corners of the space of feasible reduced forms are hierarchical mechanisms. However, one cannot immediately learn any structure about the different orderings in the hierarchical mechanisms used. The goal of this section is to show that in fact one may take all orderings in the distribution to respect the same global ordering. We begin with some definitions.

Definition 13. *A* ***hierarchical mechanism*** *consists of a function $H : \bigcup_i(T\times\{i\}) \to [k|T|]\cup\{LOSE\}$; one should interpret LOSE as a value larger than $k|T|$. On bid vector $(A_1,\dots,A_k)$, 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 $A, A' \in T$, $\pi_i(A) \geq \pi_i(A') \Rightarrow H(A,i) \leq H(A',i)$. A consequence of Border's original theorem is that the corners of the space of feasible reduced forms are hierarchical mechanisms, and therefore every feasible reduced form can be implemented as a distribution over hierarchical mechanisms. Furthermore, one can make use of results in Section A.1 to show that every feasible reduced form π can be implemented as a distribution over partially-ordered (w.r.t. π) hierarachical mechanisms. While somewhat more compelling, one might still hope that a stronger statement might be true, where each hierarchical mechanism respects not just the same partial order but a single global order. A natural hope for this global order might be to order all types by their scaled π (as in Section A.1). Unfortunately, this doesn't work, as is shown in Observation 13. Still, we show that there always exists some global ordering for which this holds: Let σ be a total ordering on the elements of $\bigcup_i(T \times \{i\})$ (i.e. a mapping $\sigma : \bigcup_i(T \times \{i\}) \to [k|T|]$). 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 $\sigma(A,i) < \sigma(B,j) \Rightarrow H(A,i) \leq H(B,j)$.

Observation 13. *There exist feasible reduced forms that are not implementable as distributions over hierarchical mechanisms that respect the global ordering of the scaled reduced form.*

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 $\pi_2(C) > 1/2$. $\square$

Theorem 22. *If a reduced form π is feasible, there exists a total ordering σ on the elements of $\bigcup_i(T \times \{i\})$ that respects π such that π can be implemented as a distribution over σ-ordered hierarchical mechanisms.*

Proof. Let σ be a total ordering over all possible types, $\sigma : \cup_i(T \times \{i\}) \to [k|T|]$. Define the unhappiness $F_\sigma(M)$ of a distribution over σ-ordered hierarchical mechanisms, M, as follows ($M_i(A)$ denotes the probability that mechanism M awards the item to bidder i when she reports type A):

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

We can view F_σ as a continuous function over a compact set: Consider the simplex whose vertices represent all σ-ordered hierarchical mechanisms Then every distribution over σ-ordered hierarchical mechanisms corresponds to a point inside this simplex. It is clear that the simplex is compact, and that F_σ is continuous in this space. Hence it achieves its minimum. Let then $M^\sigma \in \operatorname{argmin}_M F_\sigma(M)$ (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, $S_\sigma = \operatorname{argmax}_{i,A}\{\pi_i(A) - M_i^\sigma(A)\}$. 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 $MO = \operatorname{argmin}_\sigma\{F_\sigma(M^\sigma)\}$ and then refine MO as $MO_{\text{new}} = \operatorname{argmin}_{\sigma\in MO}\{|S_\sigma|\}$. 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 $M^\tau_j(A) > M^\tau_j(B) \geq 0$). 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. □

Appendix B

A Geometric Decomposition Algorithm

In this chapter we provide a poly-time algorithm for decomposing any point $\vec{x}$ in a closed convex region P into a convex combination of corners. The algorithm is folklore knowledge, but we provide it here and prove correctness for completeness. Carathéodory's Theorem states that every point $\vec{x}$ inside an d-dimensional closed convex region P can be written as a convex combination of at most $d + 1$ corners of P.

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 $\vec{z}$ satisfies with equality t non-degenerate linear equations bounding P. Hence, after at most d steps, $\vec{z}$ will satisfy with equality d non-degenerate linear equations bounding P, and therefore must be a corner, so the algorithm will terminate after at most $d + 1$ steps.

To go from step t to step $t + 1$, we pick an arbitrary corner, $\vec{a}_t$, that satisfies the same t non-degenerate linear constraints as $\vec{z}$. 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 satisfy with equality the original t linear equations as the old $\vec{z}$, as well as one new one that stopped us from further increasing c_t.

Algorithm 1 provides the formal details. In its description E denotes the set of linear equations satisfied by $\vec{z}$. The formal guarantees of the algorithm are given by Theorem 23.

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

```
1: Input: Point $\vec{x}$; separation oracle SO; bound $b$ on the number of bits required to describe
   each coefficient in the hyperplanes output by SO, as well as the coordinates of $\vec{x}$.
2: Initialize: $i := 1, \vec{y} := \vec{0}, \vec{z} := \vec{x}, E := \emptyset, c_i := 0, \vec{a}_i := \vec{0}\ \forall i \in [d+1]$.
3: Invariants: $c := \sum_i c_i$, $\vec{y} := \frac{1}{c}\sum_i c_i\vec{a}_i$, or $\vec{0}$ if $c = 0$, $c\vec{y} + (1-c)\vec{z} = \vec{x}$.
4: if $SO(\vec{x}) \neq$ yes then
5:     Output no.
6: end if
7: while $c < 1$ do
8:     Let $\vec{a}_i$ be a corner of $P$ satisfying every linear constraint in $E$. Such a corner can be found
       by solving the linear program of Figure B-1
9:     if $\vec{a}_i = \vec{z}$ then
10:        Set $c_i := 1 - c$.
11:        Output $c_1, \ldots, c_{d+1}, \vec{a}_1, \ldots, \vec{a}_{d+1}$.
12:    else
13:        Set $D := \max\{d \,|\, (1+q)\vec{z} - q\vec{a}_i \in P\}$.
14:        Set $E_i = SO((1+D+\epsilon)\vec{z} - (D+\epsilon)\vec{a}_i)$ for sufficiently small $\epsilon > 0$. /* the appropriate
           choice of $\epsilon = \epsilon(n, b)$ is explained in the proof of Theorem 23*/
15:        Update: $c_i := (1 - \frac{1}{1+D})(1-c)$, $\vec{z} := \frac{1-c}{1-c-c_i}\vec{z} - \frac{c_i}{1-c-c_i}\vec{a}_i$, $\vec{y} := \frac{c}{c+c_i}\vec{y} + \frac{c_i}{c+c_i}\vec{a}_i$, $c := c + c_i$,
           $E := E \cup E_i$, $i := i + 1$.
16:    end if
17: end while
```

Theorem 23. *Algorithm 1 correctly decomposes $\vec{x}$ into a convex combination of corners of P. The algorithm terminates in time* $\text{poly}(d, b, \text{runtime}_{SO}(d, b))$.

Proof. First, we describe how to execute Steps 13 and 14 of the algorithm, as it is clear how to execute every other step. Step 13 can be done by solving a linear program using SO. Specifically, maximize q subject to $(1+q)\vec{z} - q\vec{a}_i \in P$. For Step 14, we will explain later in the proof how to choose an ϵ small enough so that the following property is satisfied:

(P): for all hyperplanes h touching the boundary of P, and for D computed in Step 13 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 d and b.

Variables:

- $\vec{d}_i$, the corner we're looking for.

Constraints:

- $SO(\vec{d}_i) =$ "*yes*," guaranteeing that $\vec{d}_i \in P$.
- $\vec{d}_i \cdot \vec{w} = c$, for all $(\vec{w}, c) \in E$, guaranteeing that $\vec{d}_i$ satisfies every linear constraint in E.

Maximizing:

- $\vec{c} \cdot \vec{d}_i$, an arbitrary linear function to guarantee that the LP outputs a corner.

Figure B-1: A linear program to output a corner in P that satisfies every constraint in E.

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 $\vec{z} \neq \vec{d}_i$. If this is the case, then D from Step 13 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 3 hold throughout the course of the algorithm. This can be verified by simply checking each update rule in Step 15. 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 $\vec{d}_i$ and $\vec{z}$ both lie in $\bigcap_{h \in E}$ when Step 14 is executed, none of the hyperplanes in this intersection can possibly be violated at $(1+D+\epsilon)\vec{z} - (D+\epsilon)\vec{d}_i$. Therefore, the hyperplane output by $SO((1+D+\epsilon)\vec{z} - (D+\epsilon)\vec{d}_i)$ must reduce the dimension of $\bigcap_{h \in E} h$ by 1 when added to E at Step 15. Furthermore, because E_i is violated at $(1+D+\epsilon)\vec{z} - (D+\epsilon)\vec{d}_i$, but not at $(1+D)\vec{z} - D\vec{d}_i$, it must be the case that $(1+D)\vec{z} - D\vec{d}_i$ 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 d iterations of the while loop, the dimension of the remaining space is 0, and we must enter the case where $\vec{d}_i = \vec{z}$. 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

$$\sum_i c_i \vec{b}_i = (\vec{x}, 1),$$

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 closed convex region) have $\text{poly}(d, b)$ bit complexity. Applying the theory of Gaussian elimination, we deduce that each c_i can be described using no more than $\text{poly}(d, b)$ bits, so the coefficients output by our algorithm have low bit complexity. Hence the $\vec{z}$ maintained by the algorithm has $\text{poly}(d, b)$ bit complexity. So the intersections d_h of the ray $R(q) = \{(1 + q)\vec{z} - q\vec{a}_i\}$ with the hyperplanes h touching the boundary of P that do not contain both $\vec{z}$ and $\vec{a}_i$ (and hence the whole ray) also have $\text{poly}(d, b)$ bit complexity. This guarantees that we can chose ϵ to be $2^{-\text{poly}(d,b)}$ 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, d, and the runtime of SO, and each step is executed at most $d + 1$ times. So the entire algorithm runs in polynomial time. $\square$